Trabajo Fin de Grado
Grado en Ingeniería Electrónic, Robótica y Mecatrónica

Salpicadero de un monoplaza de competición

Autor: Enrique Manuel Carpintero Calderón
Tutor: Alfredo Pérez Vega-Leal

Dep. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2017

Trabajo Fin de Grado
Grado en Ingeniería Electrónoca, Robótica y Mecatrónica

Salpicadero de un monoplaza de competición

Autor:
Enrique Manuel Carpintero Calderón

Tutor:
Alfredo Pérez Vega-Leal
Profesor Contratado Doctor

Dep. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Trabajo Fin de Grado: Salpicadero de un monoplaza de competición

Autor: Enrique Manuel Carpintero Calderón

Tutor: Alfredo Pérez Vega-Leal

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2017

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Como no mencionar todo el apoyo recibido durante estos cuatro años de mi familia; de mis padres, de mis hermanas, de mis abuelos y de mi tía Mari. Siempre pendiente de mi y de que estuviera bien tanto en los malos como en los buenos momentos. Es de agradecer que cuando nada puede ir peor, estéis ahí, ahí para que todo pase a estar bien. Este proyecto también tiene un poco de vosotros. Simplemente GRACIAS.

Y como no, también, recordar a la gran familia de ARUS, la cual me permitió participar en un gran proyecto con es este y sobre todo a mis compañeros de departamento los cuales me ayudaron y enseñaron mucho.

Simplemente, para la gente que siempre está ahí sobran las palabras.

Enrique Manuel Carpintero Calderón

Ingeniero Electónico en ARUS Andalucía Racing Team.

Sevilla, 2017

Resumen

En el presente documento se detalla el diseño electrónico de un salpicadero de un monoplaza de competición. Este diseño está realizado en base a los requisitos impuestos por el equipo y por la propia normativa que regula la competición, *FSGermany2017.* Tras la evaluación de las distintas posibilidades, se ha realizado un proyecto de detalle con la opción seleccionada, incluyendo tanto el diseño hardware como el diseño software.

Tras un primer análisis de los antecedentes en este sistema en el equipo, junto con las nuevas especificaciones, se evalúan las posibles soluciones, analizando en detalle la solución escogida.

Teniendo ya la opción más viable, se hace una lista de componentes para asegurar stock antes de mandar la PCB a fabricar. Una vez escogido los componentes se realizan pruebas a los componentes más críticos con la ayuda de Arduino. Por último, en cuanto a hardware se refiere, se expondrá el proyecto de detalle, los planos y esquemas.

Una vez fabricada, ensamblada y testeada eléctricamente se pasa al diseño software, donde ayudándonos de los códigos con los que realizamos las pruebas a algunos componentes, se realiza un código capaz de cumplir con todos los requisitos.

Por último, se realiza un presupuesto y se exponen unas conclusiones junto con una propuesta de mejora.

Abstract

This document details the electronic design of a dashboard of a competition car. This design is made based on the requirements imposed by the equipment and by the regulation that regulates the competition, FSGermany2017. After evaluating the different possibilities, a detailed project has been done with the selected option, including both hardware design and software design.

After a first analysis of the antecedents in this system in the equipment, along with the new specifications, the possible solutions are evaluated, analyzing in detail the chosen solution.

Having already the most viable option, a list of components is made to secure stock before sending the PCB to manufacture. Once the components are chosen, the most critical components are tested with the help of Arduino. Finally, as far as hardware is concerned, the detail project, plans and schemes will be exposed.

Once fabricated, assembled and tested electrically, we go to the software design, where, by helping us with the codes with which we test some components, a code capable of fulfilling all the requirements is made.

Finally, a budget is made and conclusions are presented together with a proposal for improvements.

Índice

Índice de Tablas

Índice de Figuras

Notación

A^*	Conjugado
c.t.p.	En casi todos los puntos
c.q.d.	Como queríamos demostrar
■	Como queríamos demostrar
e.o.c.	En cualquier otro caso
e	número e
IRe	Parte real
IIm	Parte imaginaria
sen	Función seno
tg	Función tangente
arctg	Función arco tangente
sen	Función seno
$\sin^x y$	Función seno de x elevado a y
$\cos^x y$	Función coseno de x elevado a y
Sa	Función sampling
sgn	Función signo
rect	Función rectángulo
Sinc	Función sinc
∂y ∂x	Derivada parcial de y respecto
$x_\circ$	Notación de grado, x grados.
Pr(A)	Probabilidad del suceso A
SNR	Signal-to-noise ratio
MSE	Minimum square error
:	Tal que
<	Menor o igual
>	Mayor o igual
\	Backslash
⇔	Si y sólo si

1 INTRODUCCIÓN

La vida es 10% lo que te pasa y 90% cómo respondes a ello.

- Lou Holtz -

La *Formula Student*, también conocida como Formula SAE, es una competición entre estudiantes de universidades de todo el mundo. La Formula SAE desafía a los estudiantes a concebir, diseñar, fabricar y competir con un pequeño, pero potente monoplaza. Los equipos pasan de 8 a 12 meses diseñando, construyendo y preparando sus monoplazas para la competición. Estos vehículos son juzgados en una serie de pruebas, incluyendo una inspección técnica.

Actualmente se celebran competiciones en numerosos países como Alemania, EE. UU., Japón, Brasil, Australia, etc. Todas ellas utilizan la misma normativa base original de la Formula SAE y llegan a albergar hasta 120 equipos y más de 2.000 estudiantes. Los resultados de las competiciones son recogidos y puntúan en el ranking mundial.

El objeto de la competición es simular una situación real en la cual una empresa de competición contrata a estos ingenieros para desarrollar un prototipo. Los hipotéticos compradores serían corredores amateurs. El coche debe por ello satisfacer unas prestaciones elevadas en aceleración, frenada, y estabilidad, pero también debe ser fácil de mantener, barato, y fiable. Otros factores como la estética y el confort se valoran igualmente. El precio máximo para el vehículo es de 21.000 euros y la victoria es para el equipo que mejor logre superar todos estos requisitos. Las pruebas se pueden clasificar en eventos estáticos y dinámicos.

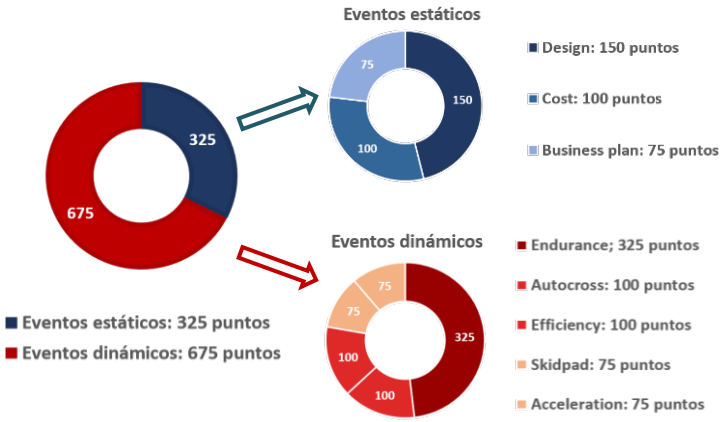

Figura 1: Pruebas y puntuaciones de la competición FSAE

El equipo ARUS *Andalucía Racing Team* es un equipo muy joven en la competición, sin embargo, sigue una trayectoria ascendente. En este año 2017, el equipo ARUS diseñó y ensambló su cuarto monoplaza. El equipo ha participado en varias competiciones en Hockenheim (Alemania), Montmeló (España) y Spilberg (Austria) con resultados que van mejorando año tras año fruto de la experiencia y el trabajo de sus integrantes.

El equipo está organizado en distintos departamentos, coordinados por el *Team Manager* y dos directores: Técnico y de Organización. Los departamentos técnicos que lo forman son Aerodinámica, Chasis, Suspensión, Motor, Interior, Electrónica y Organización y Marketing.

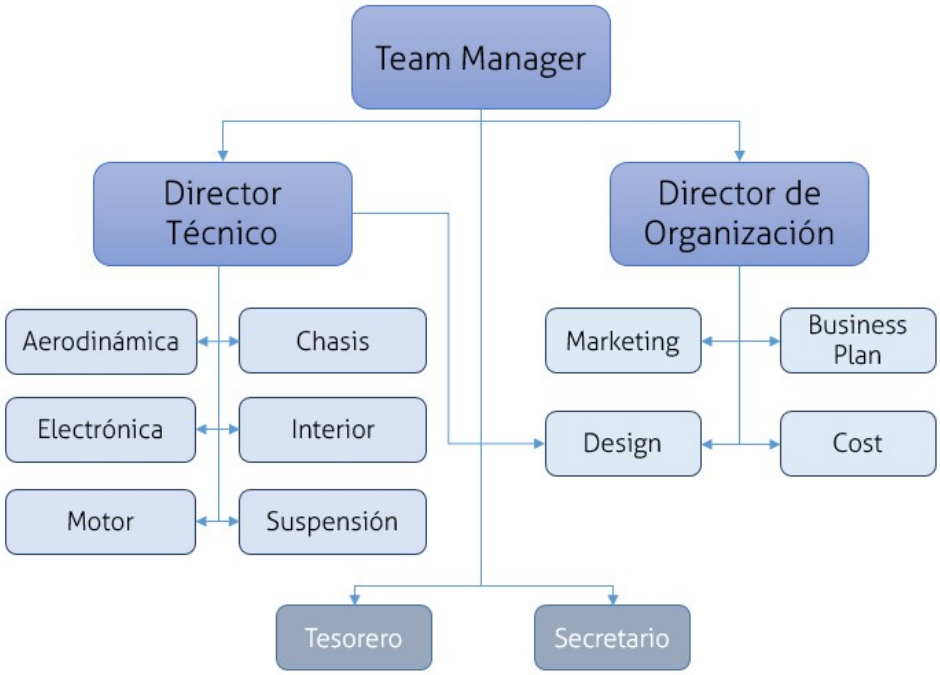

Figura 2: Organigrama de ARUS *Andalucía Racing Team.*

Este proyecto está vinculado al departamento de electrónica y en concreto al diseño electrónico del salpicadero del monoplaza ART17. La realización de este trabajo parte de la gran oportunidad que te brinda estar en un equipo universitario de competición y la posibilidad de medirte con otras universidades en las competiciones. El propósito no es otro que aprender diseñando, pero siempre con la idea de hacer algo funcional y que sea de utilidad al equipo.

Este sistema tiene el objetivo de prevenir daños irreversibles en el monoplaza, así como conocer en tiempo real una serie de parámetros adicionales que puedan ayudar a realizar un mejor set-up del coche y mejorar el pilotaje.

1.2 Requisitos iniciales

En primer lugar, hay que fijar las especificaciones que ha de tener el sistema, es decir, las prestaciones que debe ofrecer, en este caso al piloto del monoplaza mostrándole datos del vehículo en tiempo real con el fin antes mencionado de prevenir daños y mejorar el pilotaje.

- *Elementos indicadores:* pantalla LCD para mostrar valores de distintos sensores, indicador de marcha engranada, indicador de revoluciones del motor, luces indicadoras de peligro de distintas medidas e interruptores tanto para cambiar configuración de la pantalla como para interactuar manualmente con otros sistemas como la ECU o la gestión de potencia.
- *Tamaño y forma:* debe adaptarse al espacio proporcionado en el coche por el departamento de Interior a tal efecto. Esto condiciona tanto el tamaño de la PCB como el de sus componentes indicadores y actuadores. La forma debe adaptarse a la que se muestra en la Figura 3, (plano de la placa de fibra de

carbono del salpicadero a escala). Dentro de esta placa deben situarse los componentes de forma estratégica tanto para la visión por parte del piloto como para que su posicionamiento sea posible en el coche y no interactúe con otros elementos como los tubos del chasis.

- *Comunicación:* la comunicación con el resto de sistemas electrónicos del vehículo se realizará por CAN Bus, mientras que las señales críticas y las señales que provienen de los accionamientos manuales serán comunicaciones cableadas.
- *Conexiones:* La conexión general de la placa con el resto de sistemas ha de realizarse mediante un conector fiable y fácil de conectar y desconectar. A través de este conector deben pasar, además, por una parte, las señales de *Contacto* y *Arranque*, que llegan a la placa a través de otro conector y por otra las señales y alimentaciones que llegan hasta el volante, las cuales a su vez, llegan por otro conector.

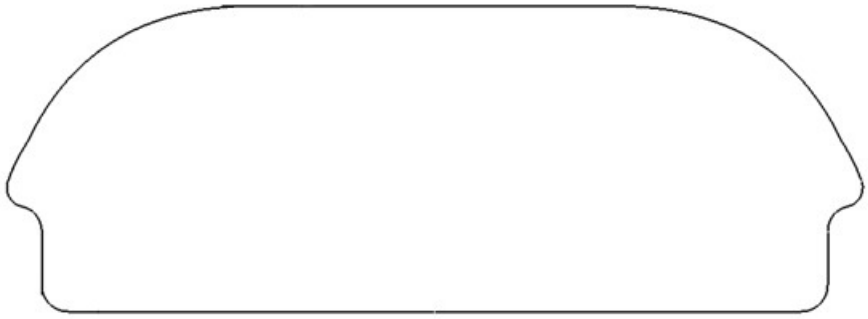

Figura 3: Forma de la placa de fibra de carbono del salpicadero.

1.3 Normativa

Las competiciones en las que el equipo ARUS suele participar son *FSAE Spain*, *FSAE Germany* y *FSAE Austria*. Todas ellas están sujetas a las normas generales de Formula Student, sin embargo, la de Alemania tiene además reglas propias para su competición, las cuales son más restrictivas que las generales. Por tanto, habrá que acogerse en todo momento a las normas de la competición alemana. Véase Referencia [1].

Pese a las estrictas normas que rigen la competición, y que están recogidas en los documentos mencionados en el anterior párrafo, en este caso no tenemos ninguna restricción por parte de la normativa a la hora de llevar a cabo los requisitos anteriores, por lo que tenemos libertad a la hora de realizar el diseño. La normativa de vehículos de combustión se centra sobre todo en aspectos de seguridad del monoplaza, como por ejemplo gestión de potencia, setas de emergencia, etc.

2 GESTIÓN Y MODO DE TRABAJO

La teoría es asesinada tarde o temprano por la experiencia.

- Albert Einstein -

En proyectos como este, el cual está vinculado a otro proyecto de mayor embergadura y al cual hay mucha gente dedicada, hay que definir claramente tanto el modo de trabajo, como la gestión de archivos con el fin de que el proyecto principal se pueda llevar a cabo sin ningún tipo de problemas. Por tanto, hay que definir bien estos dos aspectos, la gestión de archivos y documentos y el modo de trabajo.

2.1. Gestión de archivos y documentos

Con esta finalidad se ha creado una carpeta en Google Drive, donde solo los miembros del equipo pueden acceder con los permisos otorgados por el Director de Organización. Se ha elegido Google Drive porque es fácil de usar y permite usar control de versiones, es decir, se pueden recuperar todas las versiones de un archivo subido a la nube.

Cada departamento tiene su propia carpeta, todas ellas con la misma estructura. Esto es así para facilitar la navegación de cualquier miembro del equipo por todas y cada una de las carpetas ya sea esta persona miembro del departamento al que pertenece la carpeta o no. En la Figura 4 se muestra la estructura que siguen las carpetas, en este caso concreto se mostrará hasta llegar a la carpeta donde se encuentran los archivos relacionados con este proyecto.

En la carpeta “Salpicadero” se encuentra toda la información relativa a este proyecto. Es en estas carpetas donde cada responsable del proyecto al que se refiere la carpeta debe guardar toda la información relativa al mismo.

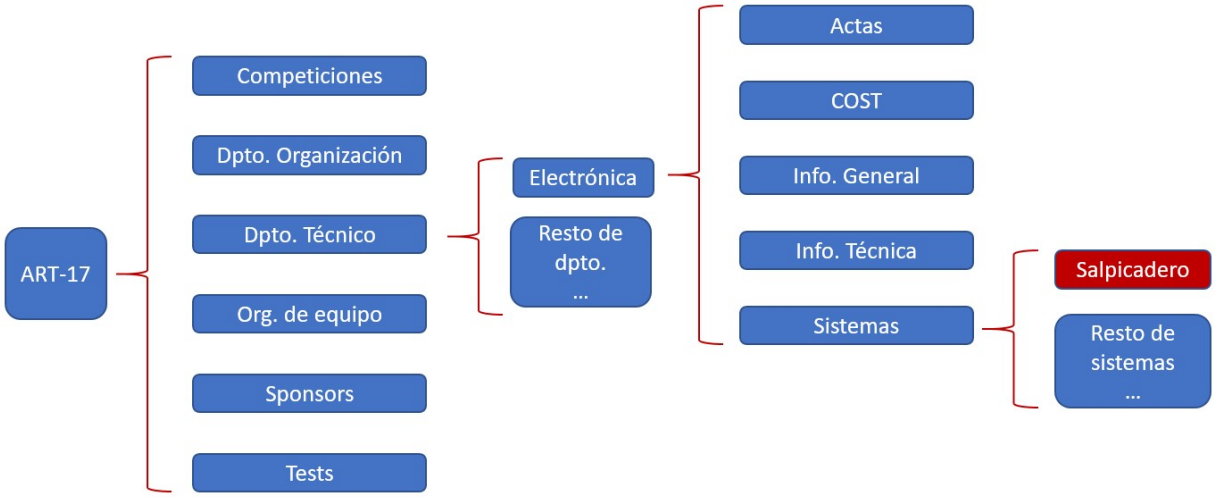

Figura 4: Organización de informacion en Google Drive.

2.2. Modo de trabajo

Al igual que pasa con la gestión de archivos, el modo de trabajo hay que estandarizarlo para facilitar el trabajo del resto de componentes del equipo y el desarrollo del proyecto. Con este fin se realizan dos tipos de reuniones semanales, la primera de ellas es la reunión de departamento y la segunda la reunión de teamleaders.

En la primera de ella, nos reuniomos todos los componentes del departamento exponiendo nuestro proyecto hasta ese momento e intercambiando información con el resto de compañeros. Esto sirve también como revisión de proyectos por parte del teamleader. Estas reuniones son concretadas a través de encuestas on-line donde cada miembro la rellena atendiendo a sus necesidades y el teamleader correspondiente la fija en el calendario con la finalidad de que cumpla con las necesidades del mayor número de personas posible

En el segundo tipo de reuniones se reúnen todos los teamleaders de cada departamento junto con el Team Manager y los directores Técnicos para exponer la situación de cada departamento y unificar los proyectos a nivel superior.

En los dos casos, se revisa que todos los sistemas cumplan con las extricta normativa para poder participar en las competiciones. Esta función en concreto, a parte de cada responsable de proyecto, es función de los directores Técnicos, los cuales exponen las deficiencias, en caso de haberlas, en la reunión de teamleader para que después sea transferido en la de reunión de departamento.

3 DESARROLLO DEL PROYECTO

La teoría es asesinada tarde o temprano por la experiencia.

- Albert Einstein -

Para la realización del proyecto se han seguido tanto la gestión, como el modo de trabajo expuesto en la Sección 2 de este mismo documento. Antes de entrar en la realización del diseño se han realizado unos estudios previos con el fin de afinar al máximo el diseño y poder por tanto cumplir las especificaciones de manera satifactoria, además, de tener la funcionalidad deseada.

3.1. Estudios previos

Antes de comenzar con el diseño se hace imprescindible un estudio previo tanto de las soluciones comerciales existentes en el mercado, como de soluciones anteriores en el propio equipo ARUS. Este estudio nos permitirá por un lado aprender de este tipo de sistemas y tener una idea de diseño y precio y, por otro lado, de las soluciones anteriores podemos, gracias al trabajo de otros compañeros, corregir los errores cometidos en anteriores diseños.

Por los motivos anteriormente comentados se van a analizar por separado los dos tipos de soluciones existentes, por un lado, las soluciones comerciales, y por otro, las soluciones de diseño propio, analizando diseños de años.

3.2.1 Soluciones comerciales

Las posibles soluciones deben ser compatibles con la ECU de nuestro coche, de la cual hablaremos a continuación. Este dispositivo, la ECU, proporciona los datos más importantes para el piloto, como pueden ser revoluciones, marcha engranada o temperatura del motor. Se han seleccionado dos de los mejores sistemas de representación de datos para este tipo de vehículos.

La ECU usada es la Link G4+ Storm, la cual se puede observar en la Figura 5. Esta es una ECU de competición, la cual permite la modificación de todos los parámetros posibles. En la Sección oportuna se hablará de la configuración del Bus CAN en este dispositivo, aspecto que se hace necesario para poder leer los datos enviados por la ECU de forma correcta.

Tabla 3–1. Comunicación ECU Link G4+ Storm

Nº de conexiones	Tipo de conexión
1	Conector 68 pines
1	Bus CAN
1	USB

En la Tabla 3-1 se muestran los diferentes puertos de comunicación que tiene dicha ECU, de los cuales el único valido para nuestra aplicación es el puerto Bus CAN.

Figura 5: ECU instalada en el monoplaza de ARUS.

3.2.1.1 Dash 2 Pro

La primera de las soluciones comerciales que cubre nuestros requisitos es el display Link Dash 2 Pro, de la misma marca que la ECU, por lo que su compatibilidad es obvia. Las características más destacadas de este dispaly se enumeran a continuación:

- Pantalla LCD visible bajo cualquier condición de luminosidad.
- Resistente al agua.
- Compacto, delgado y fácil de instalar.
- Muestra de información directamente desde la ECU usando CAN.
- LEDs de alarma programables para cualquier parámetro.
- Información mostrada configurable por software.

En la Figura 6 se muestra el aspecto de este display. Esta solución cumpliría sin problemas los requisitos impuestos, su único incombeniente es el precio, el cual ronda entre los 700€ y los 800€, el cual es totalmente inasumible por el equipo para un sistema en cierto modo prescindible.

Figura 6: Display Link Dash 2 Pro.

3.2.1.2 Haltech IQ3

La segunda de las soluciones comerciales, la cual supera los requisitos con creces, es el display Haltech IQ3 Street que es además compatible con nuestra ECU. Las características que hacen destacar este panel son las siguientes:

- Interfaz CAN compatible con más de 20 tipos de ECUs, entre la que se encuentra la Link G4+.
- Incorpora el protocolo OBDII.
- Diversos indicadores para coches de calle.
- 1 LED de alarma programable.
- Informacion mostrada totalmente programable por software.
- Función de datalogger incluida.

En la Figura 7 se puede observar el aspecto que presenta el display Haltech IQ3. Como se ha comentado anteriormete esta solución cumple los sobradamente los requisitos, pero al igual que el caso anterior su precio, que ronda los 1000€, la hace totalmente inalcanzable económicamente al equipo.

Figura 7: Display Haltech IQ3.

3.2.2 Soluciones anteriores en el equipo

La trayectoria en de este tipo de sistemas comienza en el equipo ARUS en la temporada 2015/2016, incluyendo un sistema de representación de datos en tiempo real situado en el volante. Este sistema consistía en una PCB en la que se podían encontrar dos filas de LEDs (revoluciones y alarmas), una pantalla LCD y una serie de botones pulsadores. En la Figura 8 se muestra la PCB en cuestión, de la que hablaremos en lo sucesivo para extraer de ella tanto los defectos como las virtudes de cara a poder mejorar el diseño, haciendo un sistema robusto y fiable.

Nótese que en la PCB que se muestra en la Figura 8 faltan algunos componentes, siendo el más destacable el display LCD, del que si se puede ver su conector tipo FPC.

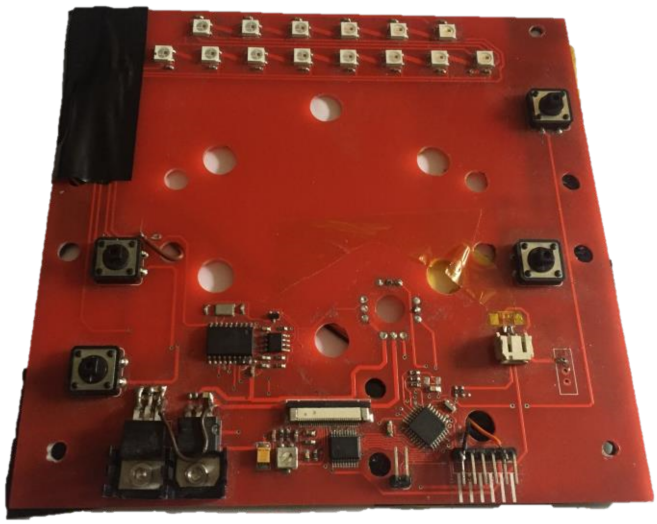

Figura 8: PCB diseñada en la temporada 2015/2016.

A las conclusiones que vamos a poder sacar a simple vista, se suman los informes de la temporada anterior indicando el funcionamiento y los posibles fallos, ya que esta PCB no terminó de funcionar debido a algunos problemas. Tengáse en cuenta de la dificultad que supone hacer una PCB totalmente funcional sin antes hacer un primer prototipo, ya que debido al presupuesto y patrocinios del equipo solo se dispone de una única oportunidad de fabricación.

Empecemos por el informe de funcionamiento de este sistema, donde se explica que el sistema funcionaba perfectamente cuando éste se encontraba situado fuera del coche, pero que dejaba de hacerlo cuando se encontraba dentro del monoplaza con el motor en funcionamiento. En concreto dejaba de funcionar la pantalla, aunque se producían cortes también en la alimentación del sistema completo. Este problema es producido por un lado al conector tipo FPC del display que debido a las vibraciones del motor no hace bien contacto eléctrico y por otro lado a la conexión entre la propia PCB y el resto de sistemas, que se hacia soldando cables directamente a la PCB. Otro de los comentarios recibidos fue que la PCB hacia cortocircuito al ponerla en contacto con la fibra de carbono del volante, por lo que tuvo que aislarse ésta tal y como se puece obsevar en la Figura 9.

Figura 9: Aislamiento de la PCB para no hacer contacto con la fibra de carbono.

Enumeremos por tanto aspectos negativos de dicho sistema:

- *Conector tipo FPC del display LCD:* este tipo de conector es el que se muestra en la Figura 10. La principal característica de estos conectores es que son muy compactos lo que hace que sus contactos tengan una superficie muy pequeña. Las vibraciones del motor del coche, que va unido directamente al chasis, pueden llegar a causar una mala conexión debido a lo comentado anteriormente.

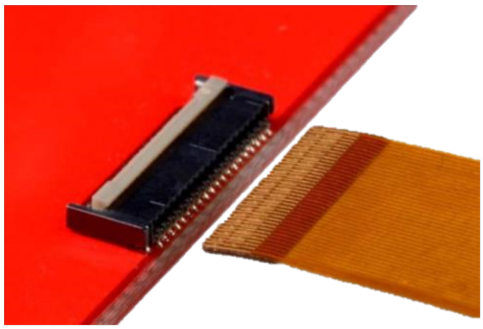

Figura 10: Conector tipo FPC.

- *Alimentación con reguladores de tensión:* el alto consumo que pueden llegar a tener los LEDs, hasta 60mA cada uno, lo que hace un total de unos 900mA hace que el regulador de tensión de 5V que alimenta el diseño, el LM7805 que puede dar 1.5A de máxima, trabaja coja una temperatura elevada, teniendo que poner unas aletas de refrigeración. Nótese que a aparte de los LEDs este regulador alimenta otros componentes del sistema.

Figura 11: Regulador de tensión LM7805.

- *Posicionamiento de la PCB sobre fibra de carbono:* como se ha comentado, la PCB iba situada en el volate, el cual está fabricado con fibra de carbono. La conductividad de este material compuesto junto con la situación de la PCB apoyada sobre este material provocó cortocircuitos en el sistema, lo cual se tuvo que solucionar de forma improvisada como puede verse en la Figura 9.
- *Conector PCB – resto de sistemas:* como se puede comprobar en las Figuras 8 y 9, donde se muestra la PCB por ambas caras, dicho sistema no tiene un conector propiamente dicho que conecte este sistema con el resto de sistemas. Esta conexión se hacía con cables directamente soldados a la PCB, estos cables llegaban al quick del volante, para poder extraer el mismo junto con la PCB. Esta situación hace que el trabajo sobre la propia PCB sea arduo debido a que no pudes extraer la PCB de forma sencilla del volante, por lo que tienes que trabajar con ella montada en el volante o desoldar los cables antes mencionados.

De igual forma, consideremos los aspectos positivos:

- *LEDs utilizados:* la principal característica de los LEDs usados en este diseño es que todos ellos se pueden controlar individualemte con una sola salida digital del microcontrolador. Este aspecto es muy interesante ya que si hubiese que controlar cada LED con una salida del micro sería totalmente inviable y habría que buscar otras soluciones, como multipexación o registro de desplazamiento.
- *Ajuste manual del contraste de la pantalla:* otro aspecto a destacar es la posibilidad de regular el contraste de la pantalla manualmente mediante un potenciometro. Este hecho facilita el ajuste de este parámetro fácilmente ya que dependiendo de la competición en la que se encuentre el monoplaza y las condiciones ambientales de ese momento se hace necesario el ajuste del mismo.
- *Botones para interactuar con el sistema:* este hecho posibilita el cambio de configuración del sistema, como, por ejemplo, la información que aparece por pantalla. Este aspecto del diseño debe ser simple y con una finalidad de configuración inicial, ya que las pruebas que se realizan en la competición son muy cortas por lo que no da para que el piloto interactúe con el sistema.

Estos aspectos mencionados, tanto positivos como negativos, se tendrán en cuenta a la hora de realizar el diseño ya que como se ha comentado anteriormente solo hay una oportunidad de fabricación, hay que basarse en hechos probados para obtener el mejor resultado posible. Y los hechos probados sobre un sistema lo más parecido posible al que se quiere diseñar y bajo las mismas circunstancias es precisamente este que acabamos de analizar.

3.2. Especificaciones y planteamiento

La solución que se ha decido tomar es la de diseñar, fabricar y ensamblar el sistema. Esta decisión se ha tomado en base al presupuesto disponible para gastar en este sistema del vehículo, junto con la flexibilidad que te da el diseñar tu propio sistema desde cero. De igual forman, se han diseñado los demás sistemas por el resto de miembros del departamento; control del embrague, adquisición de datos, gestión de potencia y telemtría. En la Figura 12 se muetran, en forma de bloque todos estos sistemas y como se conectan estos.

Como ya se comentó en el apartado “Requisitos iniciales” la comunicación debe ser por Bus CAN y es esto lo que precisamente se puede ver el la Figura 12. Nótese que aquí solo se muestran los sistemas diseñados en el departamento, junto con el transceiver, controlador de CAN, microcontrolador usados y tensión de alimetación del sistema en cuestión; por lo que falta otro elemento que está conectado al bus, que no es otro que la ECU, de la que, como ya veremos, extraemos información muy importante.

Mencionaremos aquí el funcionamiento requerido desde un punto de vista funcional, entrando más en detalles técnicos en los apartados de diseño hardware y software y el planteamiento de la solución partiendo de un esquema de la electrónica y mencionando los pasos para llevar a cabo la solución.

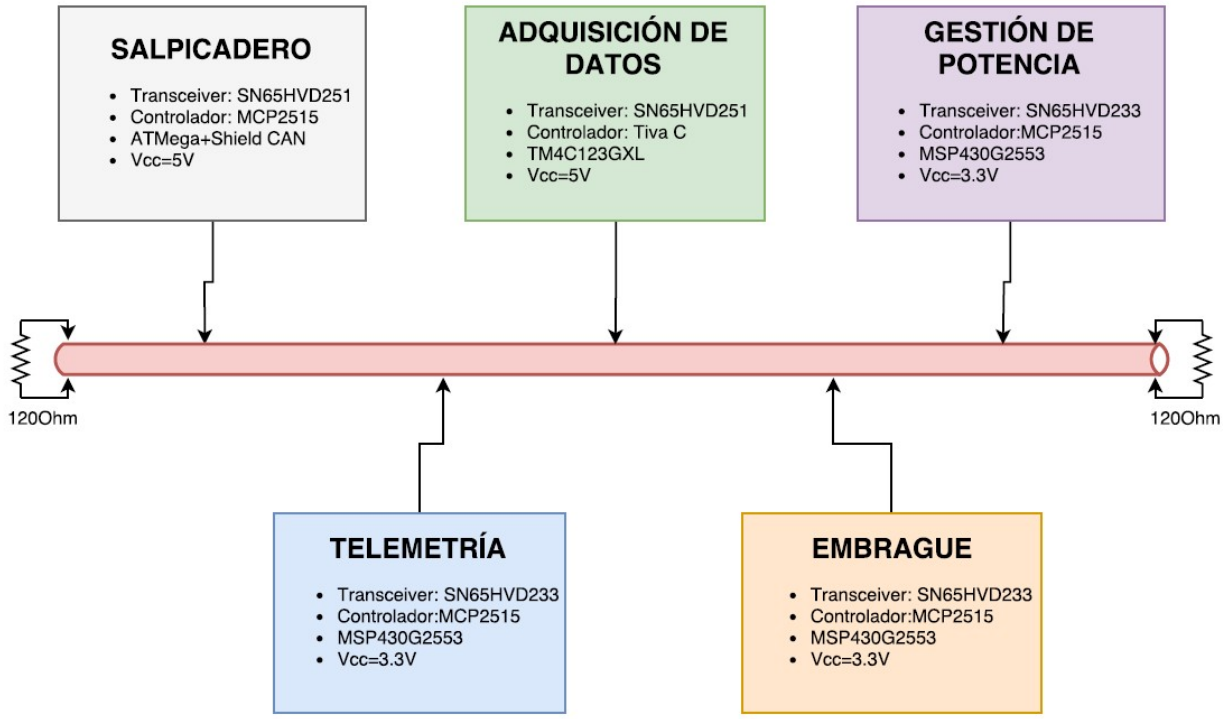

Figura 12: Esquema de conexión de los sistemas diseñados en el departamento.

3.2.1 Funcionamiento requerido

El sistema a diseñar debe tener un funcionamiento determinado, que es el que determina los componentes a incluir y por tanto el diseño. El funcionamiento que debe tener este sistema se resume en varios puntos, los cuales se exponen a continuación.

- *Datos por pantalla*: el sistema debe tener un display donde aparezcan al menos seis datos representados por un icono junto con su valor numérico.
- *LEDs de revoluciones*: las revoluciones deben ir indicadas por un conjunto de LEDs, los cuales deben ir encendiéndose en función del valor numérico de las revoluciones.
- *LEDs de alarmas*: además de la pantalla debe haber al menos seis LEDs para indicar alarmas de distintos parámetros del coche. Esto se debe a que ver una luz roja es mucho más visual e intuitivo que el propio valor para indicar una alarma.
- *Indicador marcha engranada*: La marcha engranada en la caja de cambio debe estar representada en el salpicadero. En este caso se representará en un display de 16 segmentos, ya que este permite poner también el simbono del a la marcha neutra ‘N’.
- *Botones para interactuar*: Debe haber cuatro botones de tipo palanca para poder interactuar con los distintos sistemas. Esta interactuación consiste en cambiar la configuración de la pantalla o activar o desactivar alguno de los sistemas del coche.

Y como ya se ha dicho debe adaptarse en tamaño y forma al espacio habilitado para ello, en la Figura 13 puede observarse un explosionado del salpicadero, donde aparecen todos los componentes que forman parte de él, botones, pantalla, placa fibra de carbono, tapadera trasera, etc.

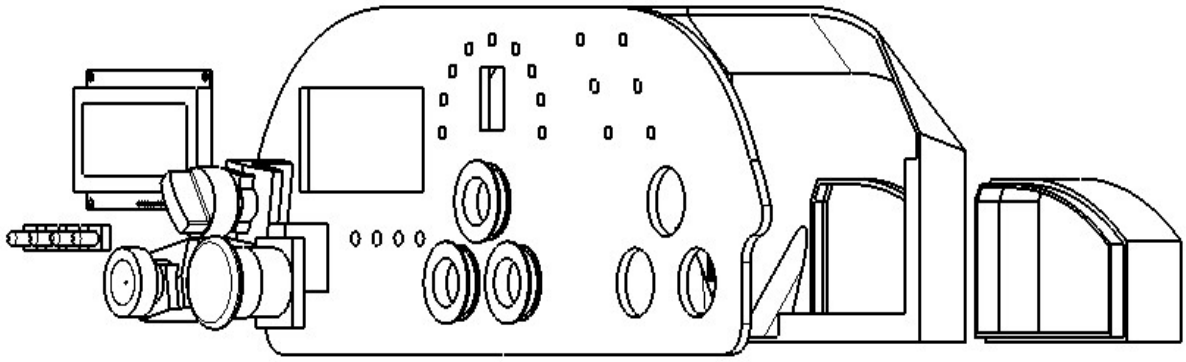

Figura 13: Explosionado salpicadero.

3.2.2 Planteamiento

Si intentamos plantear el esquema del proyecto a nivel del departamento de electrónica centrándonos en el Salpicadero llegamos al esquema representado en la Figura 14, donde se muestra a nivel de bloques todo lo que influye en el salpicadero, pero que no pertenece a este sistema, como por ejemplo otros sistemas.

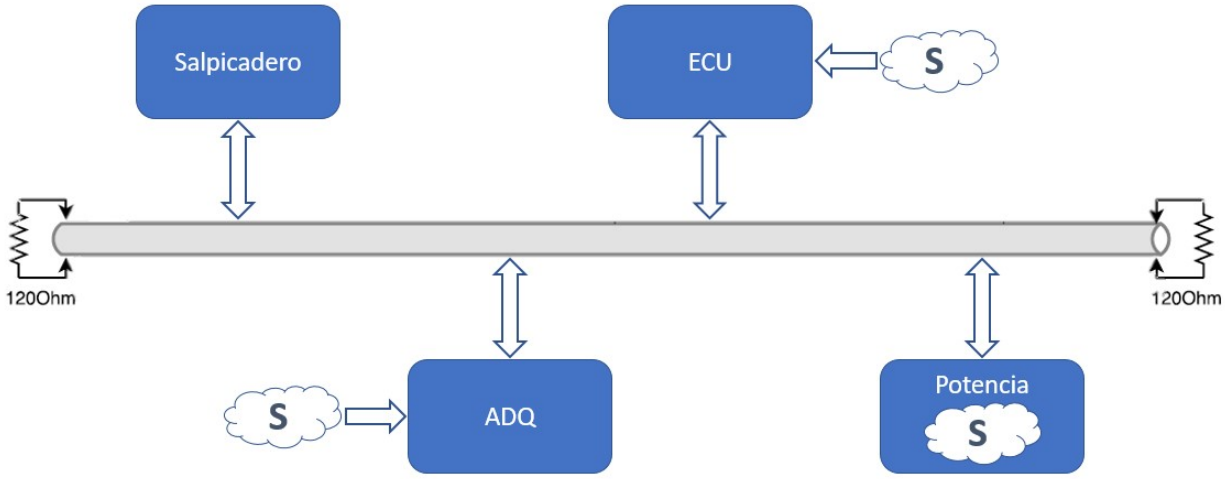

Figura 14: Diagrama de bloques del planteamiento de la solución.

En el esquema de la Figura 14 se puede observar el flujo que sigue la información desde que es captada por los sensores (nube con una 'S' en el interior) hasta que llega al sistema de salpicadero, donde son representadas. Se puede deducir facilmete que el propio salpicadero no realiza medida, sino que le llega toda la información por el Bus CAN, por lo que esta comunicación se hace de vital importancia para el funcionamiento de este sistema.

Se puede intuir que la diferencia entre que la nube que representa a los sensores esté fuera o dentro de la representación del sistema es esa misma, que el sensor o sensores se encuentren ubicados en el sistema o fuera de él, es decir, en otro lugar del coche.

Los sistemas que envían datos para ser representados en el salpicadero son la ECU, la adquisición de datos y la gestión de potencia. En la Tabla 3-1 se exponen los parámetros medidos, junto con el sistema que se encarga de su medición.

Tabla 3–2. Medidas representadas y sistema de procedencia.

Parámetro	Sistema encargado de medir
ECT o Engine Coolant Temperature	ECU
Temperatura de aceite	ADQ
Presión de aceite	ECU
Temperatura cockpit	ADQ
Nivel bateria	Potencia
Lambda	ECU
Revoluciones	ECU

Definamos cada uno de los parámetros medidos, indicando su procedencia y su magnitud. Esta será una breve definición que nos ayudará a comprender los términos antes mencionados. En la Sección correspondiente hablaremos de las unidades de medida, rango, resolución, etc de cada una de las medidas.

Nos referimos a ECT o Engine Coolant Temperature cuando hablamos de la temperatura del líquido refrigerate de un motor de combustión interna, concretamente en el punto del circuito de refrigeración donde el agua sale del propio motor. Este parámetro es medido con un sensor como el que se muestra en la Figura 15.

Figura 15: Sensor ECT.

Como su propio nombre indica, el parámetro de Temperatura de Aceite mide la temperatura del aceite del motor. Esta medida se realiza en el cárter con un sensor parecido al que mide la ECT, mostrado en la Figura 15.

De igual manera el parámetro de Presión de Aceite mide la presión del lubricante (aceite del motor) en el carter. Con este parámetro podemos ver la perdida de presión de este líquido en las situaciones reales de carrera, como por ejemplo el paso por curva, donde la presión baja. Esté parámetro se mide con un sensor parecido al que mide temperatura, en la Figura 16 podemos ver un sensor de este tipo.

Figura 16: Sensor presión de aceite.

Con Temperatura del Cockpit nos referimos a la temperatura en el habitáculo situado entre el motor y piloto, es ahí donde se encuentra el depósito de gasolina, batería, ECU, etc. Esta medida es de vital importancia debido a la proximidad de los colectores del motor y los sistemas electrónicos e inflamables que se encuentran en el cockpit. Esta temperatura se mide con el sensor LM35, mostrado en la Figura 17.

Figura 17: Sensor de temperatura LM35.

La gestión de potencia es la encargada de calcular el nivel de batería realizando una integración de la intensidad consumida y sabiendo el momento en el que la batería está al 100%, acto que se hace de forma manual. Esta medida se realiza en la propia PCB con un sensor del tipo del que se muestra en la Figura 18.

Figura 18: Sensor de intensidad.

El factor Lambda comúnmente designado con la letra griega "λ" designa la proporción aire / combustible (en peso) en forma de mezcla que entra al cilindro de un motor de ciclo Otto, comparada con la proporción estequiométrica de la mezcla ideal, de 14,7 partes de aire en peso por 1 parte de combustible en peso. Esta medida se realiza con una sonda lambda tal y como la que se muestra en la Figura 19.

Figura 19: Sonda Lambda.

Cuando hablamos de Revoluciones nos referimos a la velocidad angular del motor, normalmente medida en vueltas por minuto (RPM). Este parámetro es medido por la ECU usando dos triggers, uno en el árbol de levas del motor y otro en el cigüeñal. El sensor usado para realizar esta medida es del mismo tipo del sensor que se muestra en la Figura 20.

Figura 20: Senosor tipo trigger.

Para realizar el planteamiento del proyecto hay que tener en cuenta un factor muy importante, este factor es que solo se dispone de una oportunidad de diseño, es decir, solo se puede fabricar una PCB. Otra opción podría ser fabricar dicha PCB en el Dpto. de Electrónica de la Escuela Técnica Superior de Ingenieros de Sevilla, pero esto se hace inviable debido al tamaño y forma que debe tener el diseño. Es por este motivo por el que se ha decidido realizar el diseño teniendo como cerebro del sistema un microcontrolador programado como Arduino, esto nos permite realizar las pruebas pertinentes en las placas de desarrollo antes de hacer el diseño.

Pero antes de llegar al punto de probar componentes en la placa de desarrollo Arduino hay que recopilar toda la información recabada. Con la información obtenida en el "Estudio previo" se elabora una primera idea de diseño eligiendo los componentes a usar. Estos componentes se prueban, tal y como se ha comentado anteriormente, antes de incluirlos en el diseño para asegurar su buen funcionamiento en el sistema.

Con los componentes validados se pasa al diseño esquemático, este diseño se simula o testea si es posible. Una vez hecho esto se realiza el layout de la PCB y se manda a fabricar. Tras la fabricación, la cual se realizará por patrocinio al equipo, se realizarán una serie de tests para asegurar que ha sido fabricada correctamente. Es entonces cuando se ensambla con los componentes antes seleccionados se testea y se pasa a desarrollar el software, con el que se dará la funcionalidad deseada al sistema.

3.3. Diseño hardware

El diseño hardware tiene dos grandes partes, la eleccion y prueba de componentes y el diseño asistido por ordenador, que en este caso se realizará con *Proteus*. Para la prueba y elección de componentes nos ayudaremos de la placa de desarrollo Arduino UNO, como ya se ha comentado anteriormente, aparte de que es la placa de desarrollo de Arduino que utiliza el microcontrolador que se usará para el sistema (ATMega 328-P). Otra de las ventajas de usar Arduino es que hay gran cantidad de librerías OpenSource, lo que facilita la programación y uso de periféricos.

3.3.1 Prueba y elección de components

Los componentes más críticos y que por tanto hay que probar, son la pantalla, los LEDs y la comunicación CAN. Son los más críticos porque son lo que requieren de un diseño esquemático específico y una programación del microcontrolador para su correcto funcionamiento.

3.3.1.1 Comunicación CAN

Para implementar la comunicación CAN son necesarios un transceiver y un controlador CAN, ya que el microcontrolador que usaremos no implementa comunicación CAN. Estos componentes vienen integrados en la shield CAN de Arduino, Figura 21, por lo que no será necesario realizar montajes auxiliares, sino que simplemente hay que ensamblar la shield en la placa de desarrollo Arduino.

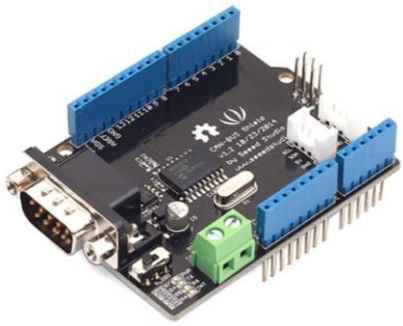

Figura 21: Shiel CAN para Arduino UNO.

Las pruebas realizadas tratan de enviar y recibir datos de otro sistema mediante el Bus CAN. Para ello es necesario programar la placa de desarrollo, y para este fin se ha utilizado la librería de Arduino que acompaña a esta shield, la cual comentaremos en la Sección Software.

3.3.1.2 LEDs

Como ya se ha comentado, los LEDs que se utilizarán serán los mismo que se utilizaron en el diseño del año pasado, es decir, el LED WS2813B. Este es otro componente crítico ya que no son LEDs normales, sino que disponen de un circuito integrado en su interior y es necesario un protocolo de comunicación determinado entre el microcontrolador y el LED para controlar el segundo. Este protocolo viene descrito en el datasheet del mismo. En la Figura 22 se muestra el montaje necesario para usar estos LEDs.

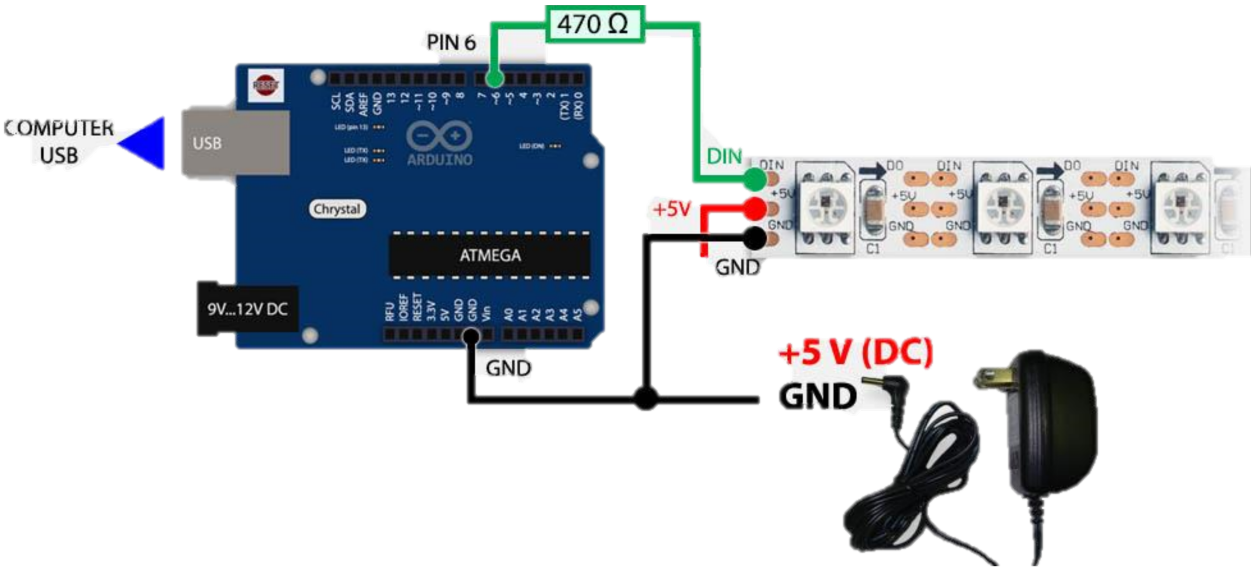

Figura 22: Motaje Arduino UNO más LEDs WS2812B.

Como puede observarse es necesario usar una fuente de alimentación externa, esto es debido al alto consumo que pueden llegar a tener los LEDs, consumo que depende del color que muestre y el brillo que estos tengan. Otro aspecto que destacar es la resistencia de aproximadamente 500Ω que hay entre la salida del microcontolador y la entrada del primer LED, esta es para evitar daños en la entrada del LED.

Las pruebas realizadas a este dispositivo es ser capaz de colorear cada LED del color que se quiera independientemente de los demás, cambiar el brillo de estos, etc. Además de probar con distintos pines de salida del microcontrolador, comprobando cuales de ellos se pueden utilizar para el control de estos LEDs.

Para la programación del microcontrolador se ha usado una librería OpenSource específica para este dispositivo de la cual se hablará en la Sección Software.

3.3.1.3 Pantalla

El componente más critico es sin duda el display LCD, por su complejidad a la hora de controlarlo. Para realizar pruebas en este compontente se ha seguido la documentación tanto del datasheet del propio display como el datasheet de la librería utilizada para programar el microcontrolador. Un aspecto imprescindible es que el display tenga controlador KS0108 o similar, que es un controlador muy común y para el que están diseñadas la mayoría de las librerías OpenSource. En la Tabla 3-3 se muestra el pinout de la pantalla según el datasheet de la misma, junto con el número de pin de Arduino al que hay que conectar cada uno, según el datasheet de la librería con la que haremos las pruebas, OpenGLCD.

La pantalla a prueba es la ERM19264-3 Series de EastRiding.

Para asegurar el funcionamiento y la capacidad de control de la pantalla se hacen diversas pruebas, como poner dibujos en pantalla, variables, etc. Nótese que para regular el contraste hay que poner un divisor resistivo o potenciómetro entre VOUT y Tierra y con salida a V0.

Tabla 3–3. Correspondencia pinout pantalla con Arduino.

Nombre	Nº terminal pantalla	Nº terminal Arduino	Descripción
D7	1	7	Dato 7
D6	2	6	Dato 6
D5	3	5	Dato 5
D4	4	4	Dato 4
D3	5	11	Dato 3
D2	6	10	Dato 2
D1	7	9	Dato 1
D0	8	8	Dato 0
E	9	Analog 4	Enable
R/W	10	Analog 2	Read/Write
RS	11	Analog 3	Data/Instruction
V0	12	-	V contraste
VDD	13	+5V	Alimentación
VSS	14	GND	Tierra
CSA	15	Analog 0	Chip Select 1
CSB	16	Analog 1	Chip Select 2
VOUT	17	-	V salida
RESET	18	RESET	Reset
LEDA	19	+5V	V+ Led
LEDK	20	GND	V- Led

3.3.2 Componentes

Una vez probados los componentes más relevantes y que pudieran dar más problemas a la hora de hacerlos funcionar, estamos en disposición de describir los componenes que se usarán para llevar a cabo el diseño. Se decribirán brevemente las características técnicas, para profundizar en ellas puede acudir a los dataheet del compoente deseado el cual puede encontrar en las referencias de este documento.

3.3.2.1 Microcontrolador ATMega 328-P

Como se ha explicado anteriormente se ha escogido este microcontrolador por dos razones principales, la primera de ellas y más importante es que este microcontrolador puede programarse fácilmente con y como un Arduino UNO y la segunda es que el equipo dispone de varias unidades de este microcontrolador de temporadas pasadas. Programar este microcontrolador como un Arduino nos da la ventaja de poder usar las librerías OpenSource, lo que hace que el diseño software sea más rápido y eficiente. En la tabla 3-4 se muestran las especificaciones de este microcontrolador.

Tabla 3–4. Resumen especificaciones ATMega 328-P

Característica	ATMega 328-P
Arquitectura	8-bit AVR
Pines	32
Memoria Flash	32 KB
Memoria SRAM	2 KB
Memoria EEPROM	1 KB
Pines GPIO	23
Puertos SPI	2
Puertos I²C	1
USART	1
ADC	10-bit 15Ksps
Canales ADC	8
8-bit Timer/Counter	2
16-bit Timer/Counter	1

En la Figura 23 se muestra el pinout del microcontrolador, donde se indica la función o funciones que puede tener cada uno de los pines que tiene el microcontrolador.

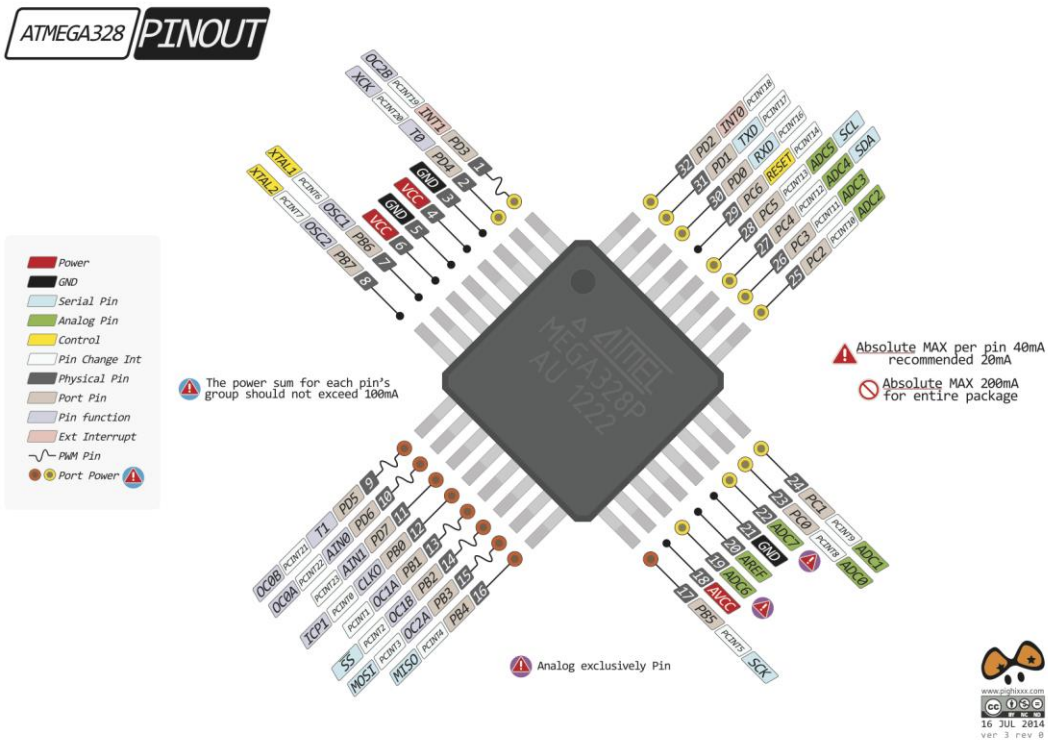

Figura 23: Pinout ATMega 328-P.

Este microprocesador ha sido desarrollado de acuerdo con el estándar internacional ISO-TS-16949, el cual contiene una serie de estrictos requisitos de resistencia a temperatura, vibraciones, etc, que lo certifica para aplicaciones automovilísticas.

3.3.2.2 Controlador CAN: MCP2515

Aunque en el Apéndice A hablaremos de todo lo relacionado con el Bus CAN, mencionaremos los componentes escogidos específicamente en este apartado, tanto del controlador como el transceiver del Bus CAN.

El controlador es el elemento encargado de la comunicación entre el microprocesador, en nuestro caso el ATMega 328-P, y el transmisor-receptor (transceiver). Este controlador se hace indispensable debido a que el microcontrolador ATMega 328-P no tiene controlador CAN incorporado. Este controlador se comunica con el microcontrolador del sistema por SPI. El aspecto es que puede observarse en la Figura 24.

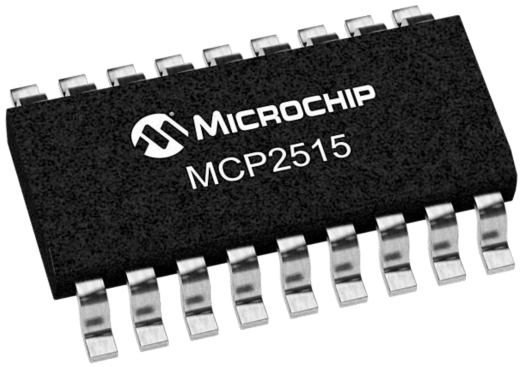

Figura 24: Controlador Bus CAN. MCP2515.

3.3.2.3 Transceiver CAN: SN65HVD251

El transceiver o transceptos es el elemento que tiene la misión de recibir y transmitir los datos, además de acondicionar y preparar la información para que pueda ser utilizada por los controladores, en este caso por el controlador MCP2515. Esta preparación consiste en situar los niveles de tensión de forma adecuada, amplificando la señal cuando la información se vuelca en la línea y reduciéndola cuando es recogida de la misma y suministrada al controlador. El transceptor se sitúa entre los cables que forman la línea CAN y el controlador. En la Figura 25 se muestra este componente.

Figura 25: Transceiver CAN. SN65HVD251.

3.3.2.4 Pantalla LCD: ERM19264-3 Series

ERM19264-3 is una pantalla monocromática con el fondo blanco y los pixeles negros, con una resolución de 192x64 pixeles. Posee una interfaz de 8 bits en paralelo y un controlador S6B0108 el cual está bien documentado. El contraste es regulable medienta una resistencia variable o una señal PWM. Tiene además un amplio rango de temperatura de funcionamiento y un amplio ángulo de visión.

Además, es fácilmente controlable por Arduino. En la Figura 26 se muestra la pantalla en cuestión.

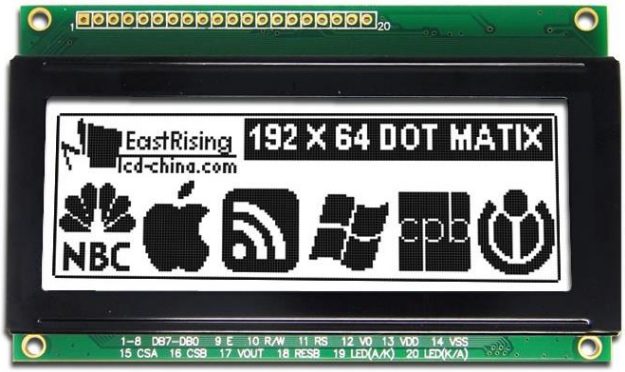

Figura 26: Pantalla ERM19264-3.

3.3.2.5 LEDs: WS2812B

WS2812B es una fuente de luz LED inteligente, en el cual el circuito de control y el LED RGB están integrados en un encapsulado 5050. Este LED como se ha comentado anteriormente tiene la ventaja de que pueden controlarse tantos LEDs como se quiera con una sola salida digital del microcontrolador. A cada LED se le puede controlar individualmente el color y en conjunto el brillo. En la Figura 27, se muestra el aspecto del LED en cuestión

Cada encapsulado tiene cuatro pines: alimentación, tierra, señal de entrada y señal de salida. Para controlar más de un LED con un solo pin se han de colocar en serie conectado la entrada de uno de ellos con la salida del anterior.

Figura 27: LED WS2812B

3.3.2.6 Convertidor DC/DC: Traco Power THL6WISM Series, 6W

Se trata de un convertidor DC/DC que transforma los 12V proporcionados por la batería del coche a los 5V necesarios para alimentar los distintos componentes de la PCB. En la Tabla 3-5 se muestran las características a destacar de este convertidor.

Sus características, junto con su tamaño compacto hace que sea el convertidor más apropiado para nuestra aplicación. Soporta además sin ningún problema las variaciones de tensión de entrada, que dependen de la carga de la batería, manteniendo la salida siempre constante a 5V.

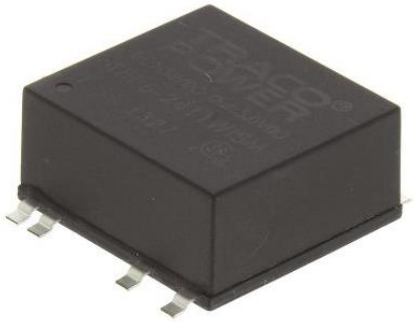

Figura 28: Traco Power. 12/5V, 6W.

Tabla 3–5. Características convertidor DC/DC.

Característica
Amplio rango de voltaje de entrada
Diseño compacto. Encapsulado SMD
Salida totalmete regulada
Aislamiento I/O 1500VDC
Rango de temp. funcionamiento: -40º hasta 75º
Protección contra cortocircuitos
Filtro de entrada según EN 55022, clase A

3.3.2.7 Otros componentes

Aparte de los componentes antes detallados, se usan otros componentes típicos de PCBs como pueden ser resistencias, condensadores, conectores, botones, etc.

Las resistencias y condensadores usados tienen el encapsulado SMD 0603, que presentan un aspecto como el que se puede ver en las Figuras 29 y 30.

Figura 29: Resistencias SMD 0603.

Figura 30: Condensadores SMD 0603.

Otro componente usado con encapsulado 0603 han sido las ferritas, las cuales se han usado como filtro de las altas frecuencias, evitando que el ruido de alta frecuencia proveniente del resto del coche entre en las pistas de la PCB.

Como se comentó, en el sistema debe haber interruptores para interactuar con el sistema. A diferencia de la PCB del año pasado, donde se usaban pulsadores, se usan interruptores de palanca ya que la función de estos será apagar y/o encender otros sistemas del vehículo y cambiar la configuración de la pantalla (de la que solo habrá dos opciones). Los interruptores usados se muestran en la Figura 32.

Para realizar conexiones con con el resto de sistemas se han usado distintos tipos de conectores, los cuales se pueden observar en la Figura 31.

Figura 32: Interruptor de palanca.

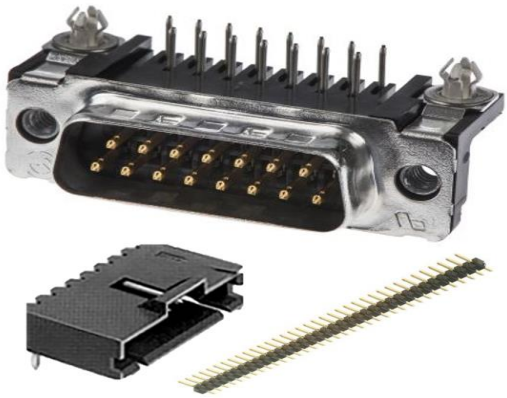

Figura 31: Conectores usados en la PCB.

3.3.3 Diagrama del Sistema

Estamos en disposición de dar forma al sistema mediante la conexión de los componentes antes mencionados, para ello nos ayudaremos de un diagrama de bloques e iremos explicando la situación de cada uno de los bloques o componentes. Este diagrama se puede ver en la Figura 33.

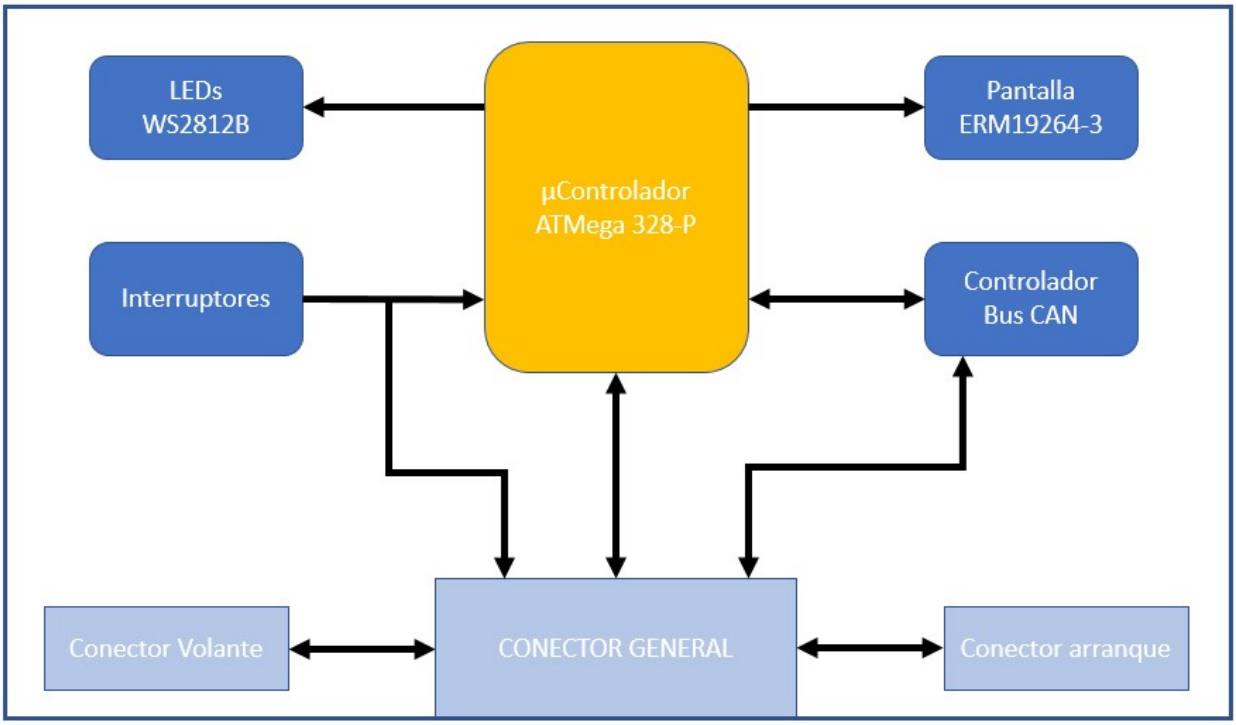

Figura 33: Diagrama de bloques del sistema.

En el centro del diagrama se encuentra el microcontrolador, cerebro del sistema, aquí llegan señales provenientes de los interruptores (cambio configuración de la pantalla), controlador CAN y alimentación del conector. De él salen señales para controlar los LEDs, la pantalla y el controlador CAN, con el que se comunica por SPI.

Al conector general llegan señales que vienen del controlador CAN (Bus CAN), de los conectores de volante y arranque e interruptores. La señal de los interruptores va directamente al sistema al que afectan, como por ejemplo la señal del interruptor que cambia el mapa motor va direcramente a la ECU.

Si nos damos cuenta aquí no aparece nada del display siete segementos donde se representa la marcha engranada

en la caja de cambios, el cual era un requisito de funcionamiento, esto es debido a la insuficiencia de pines de salida del microcontrolador. Por lo que este requisito lo deberá proporcionar el otro sistema del coche que está situado en la misma PCB, la Telemetría.

En el Apéndice A, se muestra el diseño esquemático completo, donde se pueden ver todas y cada una de las conexiones que antes se han mostrado en forma de diagrama de bloques, ya que además de estos componentes principales es necesario otros muchos para el acondicionamiento de señales y componentes auxiliares a estos compontes, como condensadores de alimentación, resistencias terminales, etc.

3.3.4 Diseño asistido por ordenador

Con la idea de diseño totalmente definida se pasa a realizar el diseño definitivo, para ello nos ayudamos de herramientas de diseño asistido por ordenador, en concreto para este proyecto usaremos Proteus. Proteus Design Suite es una aplicación para la ejecución de proyectos de construcción de equipos electrónicos en todas sus etapas: diseño del esquema electrónico, programación del software, construcción de la placa de circuito impreso, simulación de todo el conjunto, depuración de errores, documentación y construcción. Consta de dos programas principales: ARES e ISIS.

El programa ISIS (Intelligent Schematic Input System) permite diseñar el plano eléctrico del circuito que se desea realizar, con componentes muy variados, desde simples resistencias hasta microcontroladores o microprocesadores, incluyendo fuentes de alimentación, generadores de señales y muchos otros componentes.

ARES (Advanced Routing and Editing Software) es la herramienta de enrutado, ubicación y edición de componentes que se utiliza para la fabricación de placas de circuito impreso. ARES cuenta con poderosas herramientas para el posicionamiento y rutado automático de componentes, herramienta que no se puede usar en nuestro diseño debido a que cada componente tiene una ubicación definida de antemano en la placa.

Figura 34: Proteus Design Suite.

Otra de las herramientas que tiene Proteus es que te permite ver los diseños en tres dimensiones, en las Figuras 35 y 36 se puede ver el diseño definitivo de la PCB del salpicadero y telemetría del monoplaza ART-17.

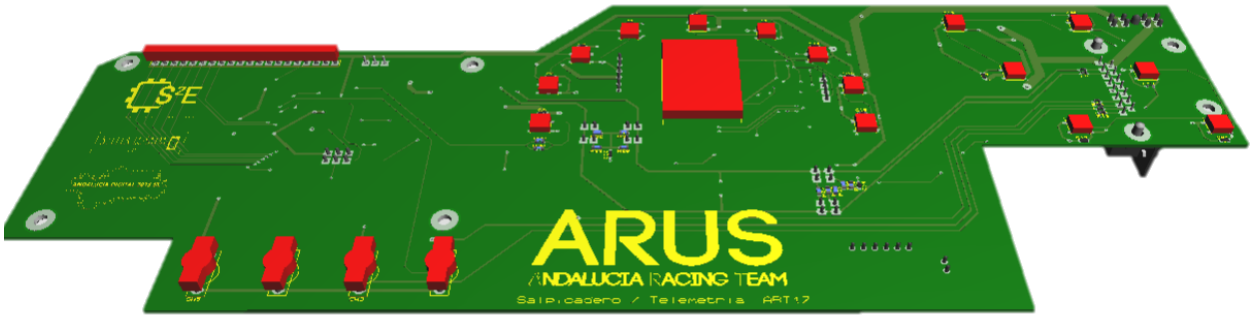

Figura 35: Placa circuito impreso vista 3D. Vista Top.

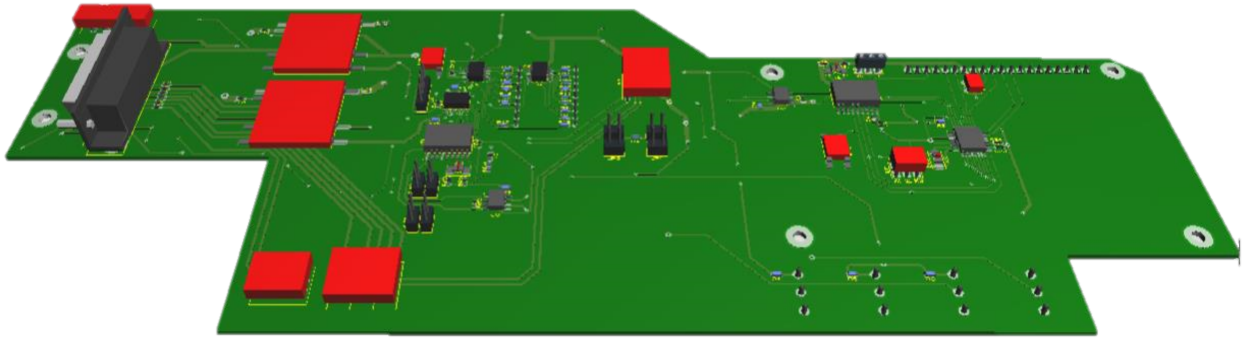

Figura 36. Placa circuito impreso vista 3D. Vista Botton.

La extraña forma de la placa se debe a que como se comentó en la Sección "Requisitos iniciales" tiene que adaptarse a la forma de la placa de fibra de carbono del salpicadero. La forma de esta placa de fibra de carbono con la ubicación exacta de los componentes se muestra en la Figura 37. La PCB va situada detrás de esta placa, por lo que los componetes deben coincidir perfectamente con el espacio reservado para ellos.

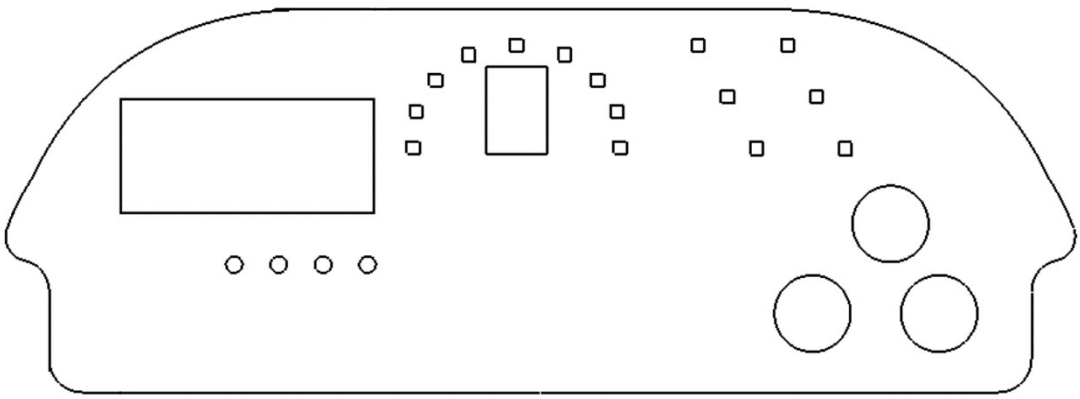

Figura 37: Plano placa fibra de carbono del salpicadero.

Si nos fijamos en la Figura 35 la PCB tiene en la parte más baja dos espacios libres, estos son para permitir la ubicación de los tres botones de arranque que se pueden ver en la Figura 36 abajo a la derecha y el hueco que deja la PCB al lado izquierdo es para permitir la colocación del mando del repartidor de frenada en caso de que

el departamento opotuno desee instalarlo.

En los Apéndices A y B, se puede consultar la totalidad del diseño, tanto el diseño esquemático tanto el rutado de la placa de circuito impreso, por capas y completo.

3.4. Diseño software

Una vez diseñado el Hardware, con un resultado como el que se puede apreciar en las Figuras 35 y 36, se pasa al desarrollo software, para el cual, como ya se ha dicho, nos ayudaremos de las librerías OpenSource disponible para Arduino. La decisión de programar en Arduino se toma en base, precisamente, a eso, a las librerías disponibles para los periféricos usados, facilitando la programación de manera considerable.

3.4.1 Entorno de desarrollo

Como se acaba de comentar, microcontrolador se programará como un Arduino y para este propósito nos ayudaremos del IDE de Arduino. IDE (Integrated Development Environment) es un entorno de programación compuesto por diferentes herramientas que ayudan al desarrollador a crear determinadas aplicaciones. Se compone de un editor de código fuente, un compilador, depurador e interfaz gráfica. Esta IDE tiene, además, barra de herramientas, editor de código, terminal de mensajes del compilador y monitor serie entre otras.

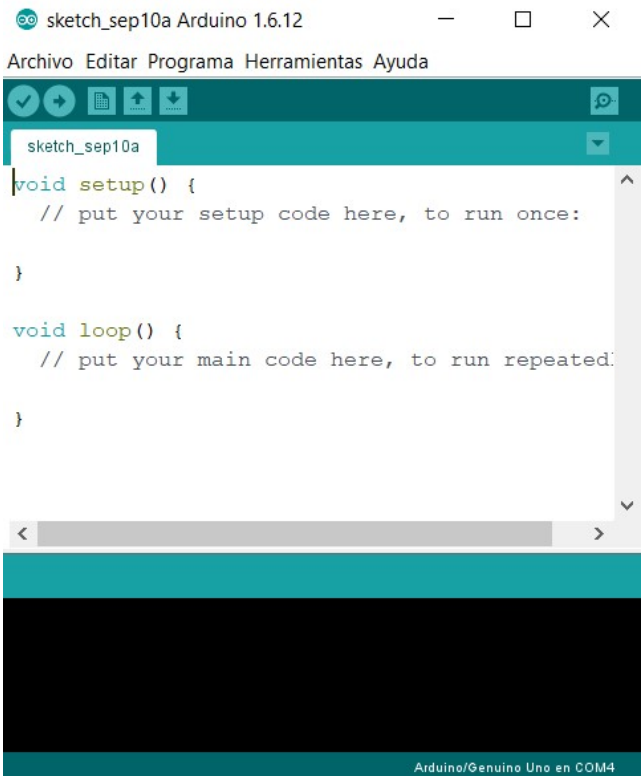

Figura 38: Entorno de programación Arduino.

Para cargar un programa al microcontrolador en la placa de desarrollo, debe conectarse el dispositivo al ordenador vía USB tal y como se muestra en la Figura 39. Esto solo se puede hacer para subir programas a placas de desarrollo Arduino, las cuales tiene una serie de circuitos integrados que pasan de comunicación USB a comunicación serie para poder transferir el programa al microcontrolador. Este no es nuestro caso, ya que en la PCB solo tenemos el microcontrolador, sin circuitos integrados auxiliares, por lo que la forma de programar nuestro sistema es diferente y se explica en la Sección 3.4.4.

Figura 39: Conexión USB entre la placa Arduino y un PC.

3.4.2 Librerías

Como se ha comentado en varias ocasiones, una de las razones por la que se ha decidido usar el microcontrolador ATMega 328-P es que se puede programar como un Arduino y la ventaja de que se pueda programar como un Arduino es la cantidad de librerías OpenSource que se pueden usar. En la Tabla 3-6 se expone el nombre de la librería junto con el periférico que ayuda a controlar y una pequeña descripción.

Tabla 3–6. Librerías OpenSource Arduino.

Libreria	Periférico	Descripción
Adafruit NeoPixel	LEDs WS2812B	Control de los LEDs
CAN BUS Shield Master	Comunicación CAN	Funciones CAN
openGLCD	Pantalla LCD	Uso de la pantalla
U8glib	Pantalla LCD	Uso de la pantalla
TimerOne	Timer interno	Uso y configuración del Timer
SPI	Comunicación SPI	Uso y configuración del SPI (Bus CAN)

Como se puede observar hay dos librerías que sirven para controlar el mismo dispositivo, estas son *openGLCD* y *U8glib*, las cuales sirven para el control y manejo de la pantalla, pues bien, esta mención se debe a que hubo un problema durante el dasarrollo software con una de ellas, esto lo comentaremos con mas detalles en la Seccíon de *Problemas y soluciones*.

3.4.3 Programación y código

En el siguiente diagrama de flujo, Figura 40, se puede ver de forma muy esquematizada el programa que debe correr el microcontrolador, si entrar en detalle en qué hace cada una de ellas.

El flujo del diagrama depende de dos factores principalmente, de que llegue un mensaje por el Bus CAN y de que el temporizador sature y salte la bandera *Refresh* para refrescar la pantalla.

Nótese de nuevo que aquí no aparecen todos los detalles del programa, ejemplo de ellos el interrupotor de cambio de configuración de la pantalla, entrada que se monitoriza en los bloques del diagrama donde pone “Actualizar pantalla”.

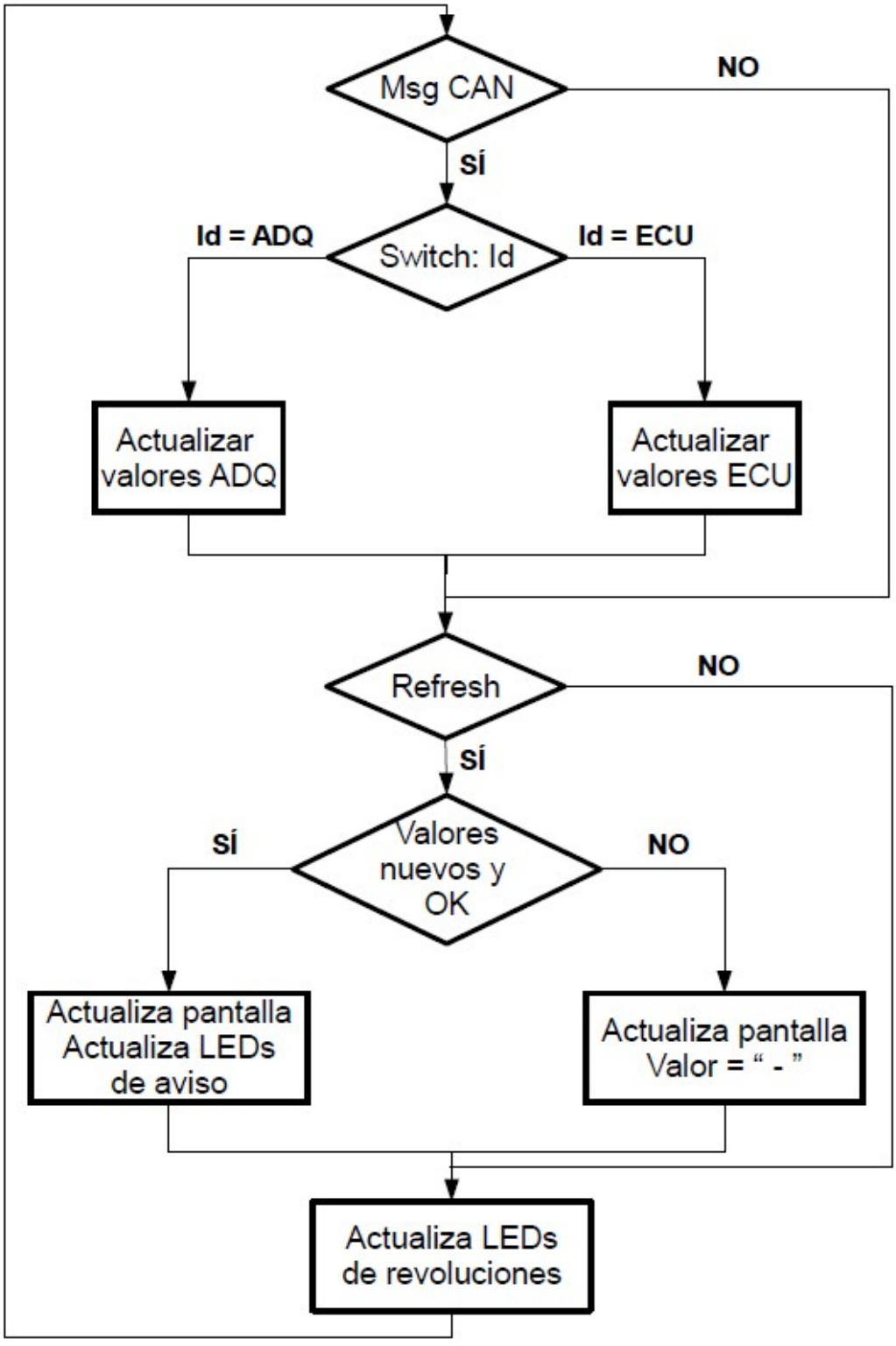

Figura 40: Diagrama de flujo programa principal.

Como ya se ha comentado y se puede comprobar en el diagrama, todo depende de una buena comunicación con el resto de sistema, la cual se realiza a través del Bus CAN. Para dar forma a esta comunición dentro del coche se ha creado la Tabla 3-7, la cual contiene toda la información necesaria para identificar mensajes y datos en el Bus CAN.

Tabla 3–7. Protocolo CAN ART-17.

Sistema	Id (dec)	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Frecuencia
ECU	**70**	Revoluciones		ECT	Lambda		TP	Presion Aceite		50Hz
ECU	**40**	IAT	MAP	ECT	Lambda		Marcha	-	-	10Hz
Potencia	**50**	Int 1	Int 2	Int 3	Carga	Voltaje	-	-	-	50Hz
ADQ	**90**	WS FR		WS FL		WS RR		WS RL		50Hz
ADQ	**80**	G. volante	Ax	Ay	Az	Ext FR	Ext FL	Ext RR	Ext RL	1Hz
ADQ	**20**	Record	Tª aceite	Tª cockpit	Tª rad.	Fr. R	Fr. L	Carga	-	1Hz

Las tramas de interés para el salpicadero son las tramas cuyo identificador es igual a 70 ó a 20, es decir, una que viene de la ECU y otra que viene de la ADQ. De la primera de ellas obtenemos los valores: Revoluciones, ECT, Lambda y Presión de aceite, mientras que de la segunda obtenemos: Record, Temperatura de aceite, Temperatura del cockpit y Carga de la batería.

3.4.3.1 Acondicionamiento de señales

Estos valores requieren de un acondicionamiento antes de ser mostrado por pantalla, además de una valización para que en caso de fallo electrónico no se muestren por pantalla datos desorbitados que den lugar a la confusión del piloto.

En la siguiente Tabla se describe el acondicionamiento y otra serie de aspectos a tener en cuenta.

Tabla 3–8. Acondicionamiento de señales.

Id (dec)	Byte(s)	Parámetro	Mínimo teorico	Máximo teorico	Mínimo real	Máximo real	Unidades	Conversión
70	2	Revoluciones	0	65535	0	13000	Rpm	A*256+B
70	1	ECT	0	255	0	200	ºC	1
70	2	Lambda	0	65535	0	1.5	-	(A*256+B)/1000
70	2	P. aceite	0	65535	0	500	bar	A*256+B
20	1	Tª aceite	0	255	0	255	ºC	1
20	1	Tª cockpit	0	255	0	150	ºC	1
20	1	Carga bat.	0	255	0	100	%	1

Estos valores se representan de la siguiente forma:

- Las revoluciones se representan de forma línea en 9 LEDs, de los que se encienden una parte proporcional de ellos. El número de LEDs que se encienden es la parte entera del número que resulta al hacer una regla de tres entre 0 y 13000.
- ECT (Engene Coolant Temperature) se representa de forma numérica en la pantalla LCD, donde se representan las unidades de esta medida. Tiene una resolución de unidades.
- El factor Lambda se representa en la pantalla de forma númerica, representado su unidad y sus milésimas. Tiene una resolución de milésimas.

- La Presión de aceite se representa númericamente en la pantalla, representando sus únicamente sus unidades. Tiene una resolución de unidades.
- La Temperatura de aceite se representa de forma numérica en la pantalla LCD, donde se representan las unidades de esta medida. Tiene una resolución de unidades.
- La Temperatura del cockpit se representa de forma numérica en la pantalla LCD, donde se representan las unidades de esta medida. Tiene una resolución de unidades.
- La Carga de la batería se representa en la pantalla de forma porcentual. Tiene una resolución de unidades.

3.4.4 Quemar bootloader y subir programas

Cuando cargamos un programa en Arduino desde el USB con el IDE, estamos haciendo uso del bootloader, se trata de un pequeño programa que ha sido guardado previamente en el microcontrolador de la placa y que nos permite cargar código sin necesidad de hardware adicional. El bootloader solo está activo unos segundos cuando se resetea el Arduino y después comienza el sketch que está cargado en la memoria flash de Arduino y que hemos programado y subido a la placa.

El bootloader se ejecuta cuando el microcontrolador se enciende o se pulsa el botón reset, durante un corto espacio de tiempo espera que le llegue por el puerto serie un nuevo sketch desde el IDE de Arduino (este distingue un sketch de otra cosa porque tiene un formato definido). Si llega un sketch, este es guardado en la memoria flash y ejecutado, en caso contrario ejecuta el sketch anteriormente cargado.

Por lo que el primer paso que hay que realizar es quemar el bootloader del microcontrolador ensamblado en la PCB, para permitir la subida de programas posteriormente desde Arduino IDE.

Para ello vamos a usar los pines destinados a ICSP (In Circuit Serial Programming), que no son más que los pines del puerto SPI más Reset, Alimentación y Tierra. Con este fin se ha puesto fácil el acceso a estos pines mediante jumpers. Para llevar a cabo la quema del bootloader es necesario realizar el montaje que se muestra en la Figura 40, suponiendo que la placa de la derecha es nuestro sistema con el microcontrolador ATMega328-P.

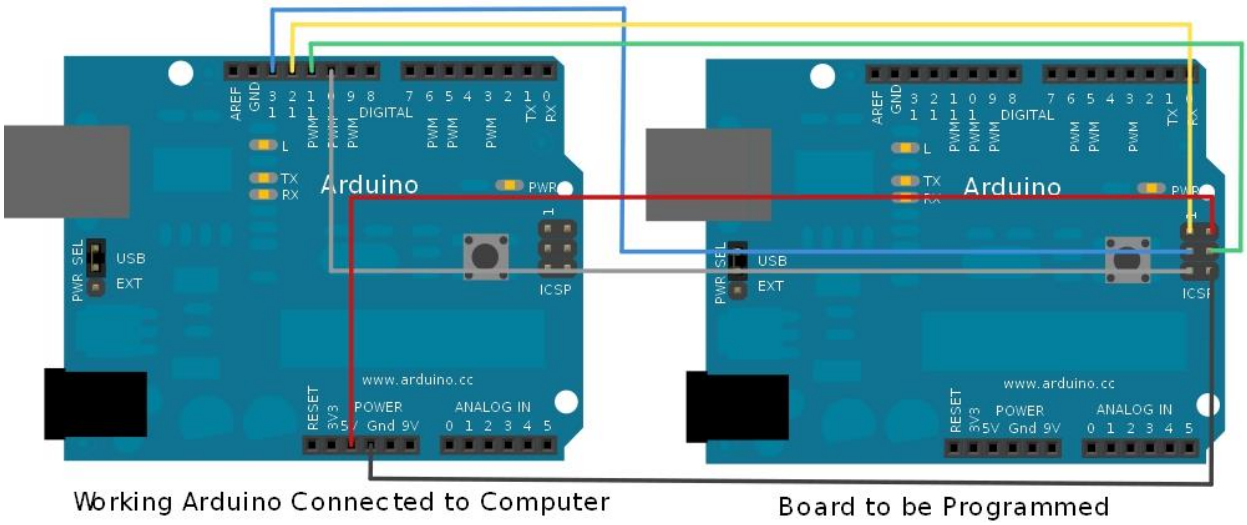

Figura 41: Montaje para programar microcontrolador en PCB.

La placa de desarrollo de la derecha se conecta al PCB tal y como se muestra en la Figura 38 y se le carga el programa que viene como ejemplo en Arduino IDE “ARDUINO ISP”, Figura 42.

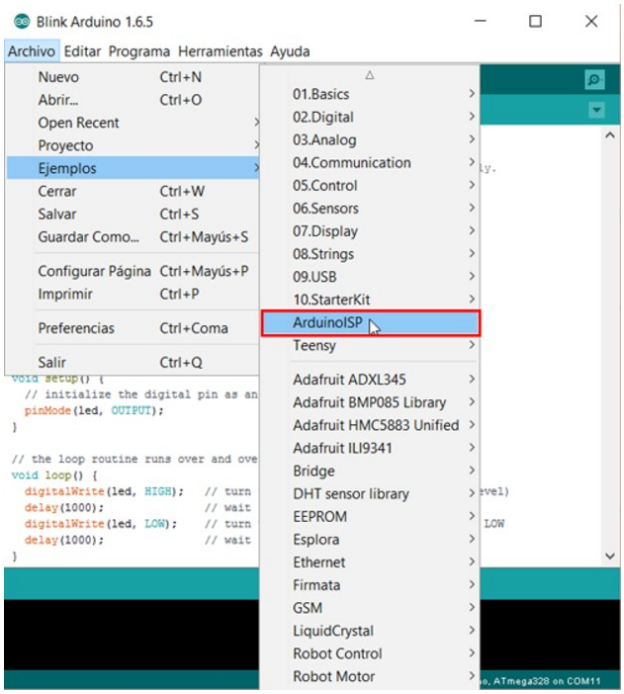

Figura 42: Programa "ARDUINO ISP".

Seguidamente debemos cambiar el programador en *Herramientas > Programador > Arduino as ISP*, tenemos que cambiar también el tipo de Arduino que queremos programar y el microprocesador: *Herramientas > Placa > Arduino UNO y Herramientas > Procesador > ATmega328*. Por lo que el aspecto final debe ser parecido al de la Figura 43.

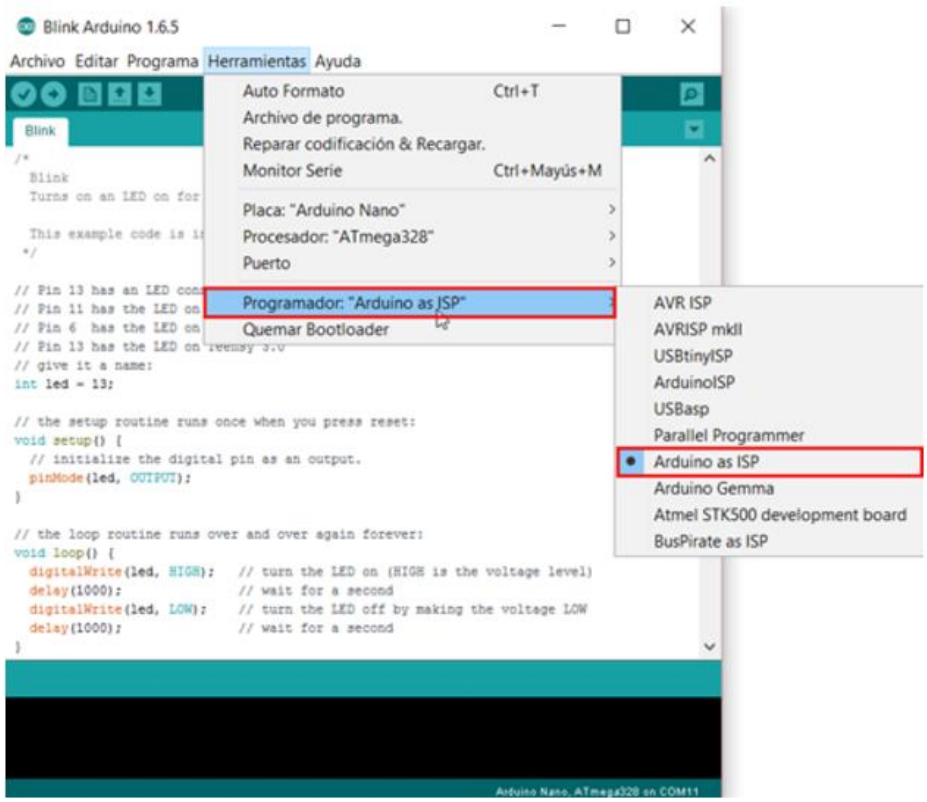

Figura 43: Aspecto final de la configuración.

Por último, para quemar el bootloader solo hay que picar en la opción: *Herramientas > Quemar Bootloader*.

Para subir un programa al microcontrolador que se encuetra en la PCB la configuración es la misma, el único cambio se produce en el último paso, donde en vez de Quemar Bootloader, hay que picar en la opción *Programa>Subir usando programador*, ver Figura 44.

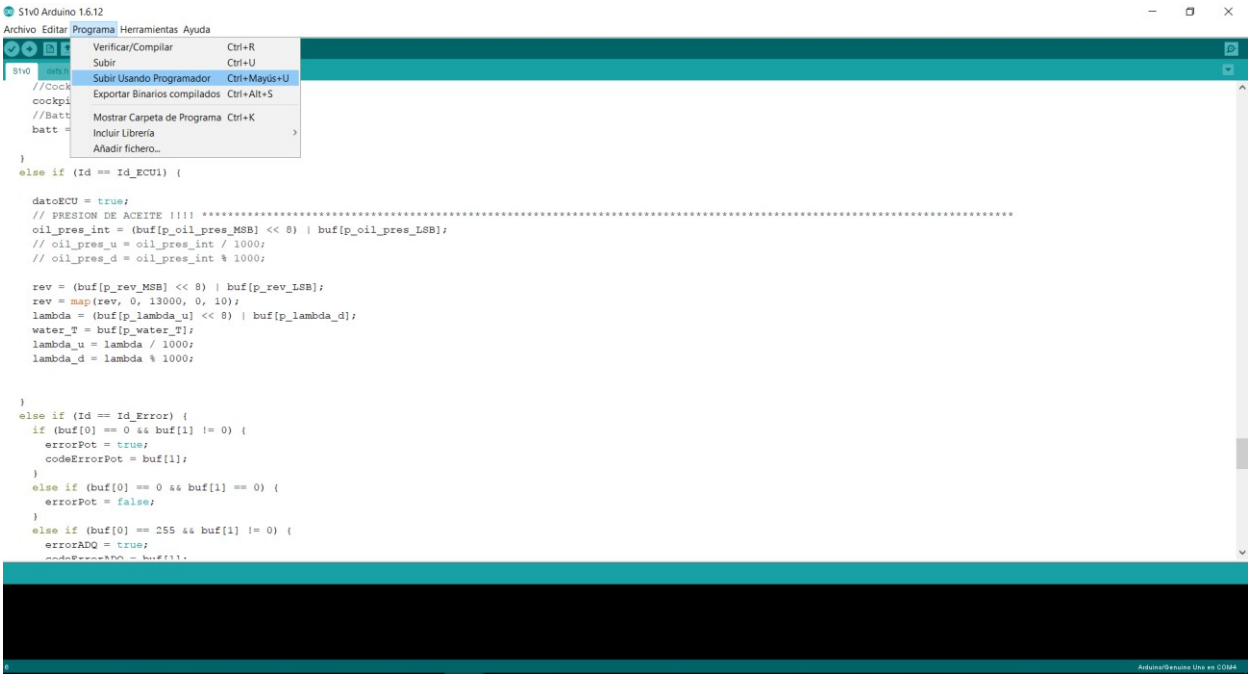

Figura 44: Subir programa a microcontrolador en PCB desde Arduino IDE.

3.4.5 Configuración de la ECU

Como se comentó cuando hablamos sobre la ECU, los parámetros de esta son totalmente configurables. Para que cumpla con el propósito que seguimos debe seguir el protocolo marcado en la Tabla 3-7, por lo que debemos configurar dicho dispositivo. En la Figura 44 se muestra la interfaz gráfica que facilita Link para este fin.

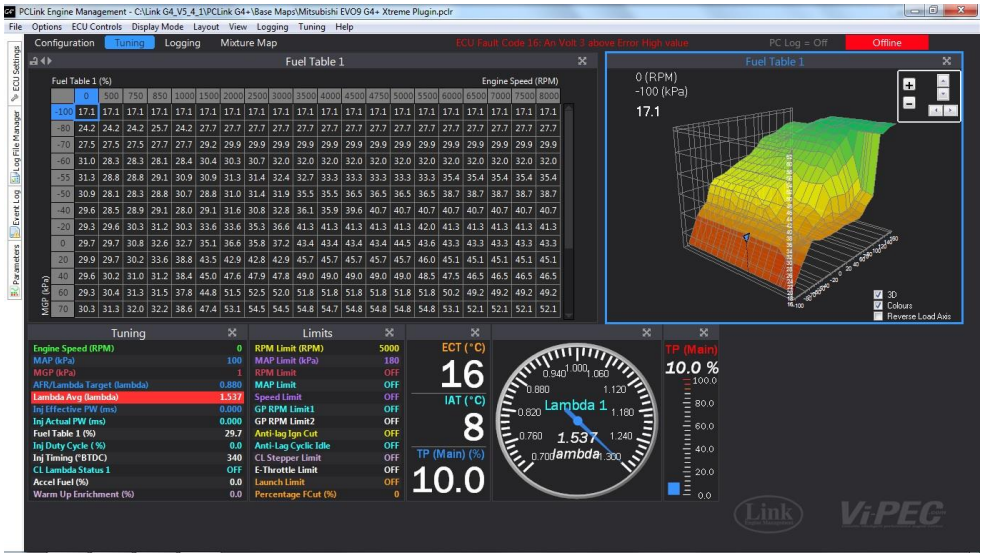

Figura 45: Interfaz gráfica Link.

Debemos ir a la pestaña que se muestra en la Figura 45 para configurar todos los aspectos relacionados con el Bus CAN. A esta pestaña podemos llegar a través de la ruta *Options* > *CAN Setup.* Aquí podemos personalizar las tramas que se transmiten, pudiendo elegir qué parámetros se envían, en que órden y con qué frecuencia.

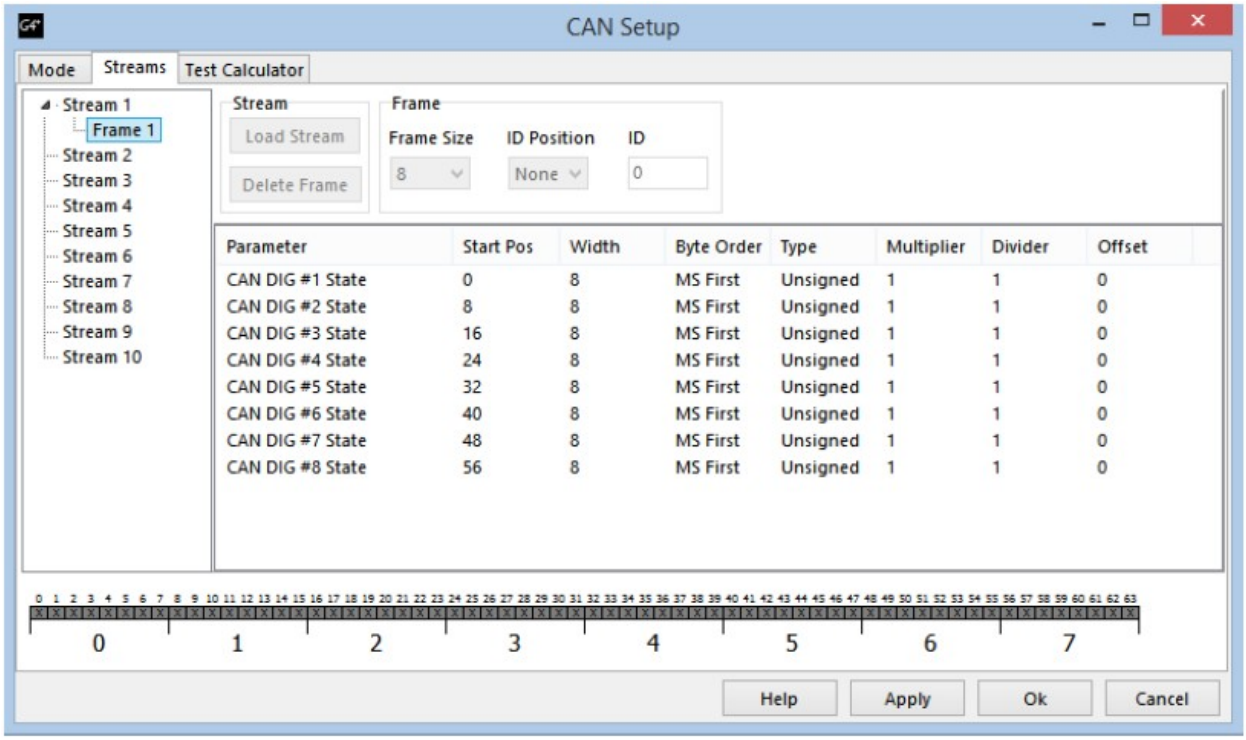

Figura 46: ECU Link, CAN Setup.

4 PROBLEMAS Y SOLUCIONES

El fracaso es, a veces, más fructífero que el éxito.

- Henry Ford -

4.1. Conflicto entre la libreria openGLCD y SPI

Como se explicó en la *Sección de Prueba y elección de componentes* se probó la pantalla con la librería *openGLCD*, pero esta prueba se hizo independientemente de otros componetes.

Cuando se intentó programar la PCB usando a la vez el SPI para la comunicación con el controlador CAN con la pantalla, esta no funcionaba correctamente y este error deriva de que una de las señales de datos de la pantalla viene del puerto de Arduino que tiene la función de CS (Chip Select) del bus SPI. Esta acción es posible ya que para esta aplicación no es necesaria la señal CS del SPI. Tras varias modificaciones en la librería *SPI* y *openGLCD* el problema persistía.

Finalmete el problema fue solucionado usado como librería para el control de la pantalla LCD la librería *U8glib.* Esta librería tiene un funcionamiento similar a la anterior, con la ventaja de que aquí tú puedes definir la función de los puertos de Arduino con una simple definición en el código.

Esta es una librería, al igual que las demás, OpenSource que sirve, además, para el control de displays con distintos controladores.

4.2. Puertos de salida insuficientes

En los requisitos de funcionamiento se incluye un display siete segmentos el cual no se incluye después en el diseño, esto es debido a que no hay pines suficientes para el control de este dispositivo.

Tras valorar el cambio de microcontrolador, para incrementar el número de puertos de salida disponibles se decide que el control de este dispositivo se encargue el otro sistema que se encuentra ubicado en la PCB, la Telemetría, cuyo microcontrolador tiene la capacidad necesaria para realizar el control de este display.

Esta decisión se toma en base a que se dispone del microcontrolador que vamos a usar, por lo que cambiar de microcontrolador supone un gasto adicional, además de que hay otro sistema en la misma PCB el cual se puede encargar de mostrar las marchas por el display sin ningún tipo de problemas.

5 PRESUPUESTO

No basta tener buen ingenio, lo principal es aplicarlo bien.

- René Descartes -

En la siguiente Tabla se detalla el presupuesto del proyecto, atendiendo a la mano de obra ya que los materiales corren por parte del equipo.

Tabla 5–1. Presupuesto.

Descripción	Cantidad (h)	Precio (€/h)	Precio total
Diseñador electrónico	55 h	30€/h	1.650 €
Programador	40 h	40 €/h	1.600 €
Técnico rutado	25 h	20 €/h	500 €
Ingeniero	180 h	30 €/h	5.400 €

TOTAL 9.150 €

APÉNDICE A. ESQUEMÁTICO

Microcontrolador y elementos auxiliares

En la Figura 47 se muestra el diseño esquemático del microcontrolador, además del conector de programación (J4), el condensador de desacoplo en la alimentación (C28), el oscilador con su respectivo circuito y el pulsador de resete con su resistencia de pull up.

En las siguientes Figuras que muestran el esquemático se podrá ver donde va conectado cada pin del microcontrolador. El conector de programación es necesario, como su propio nombre indica, para hacer posible la programación del microcontrolador. En cuanto al pulsado de reset, lo único que cabe decir de él es que tiene una resistencia de pull up porque esta señal de entrada, la de reset, es activa a nivel bajo.

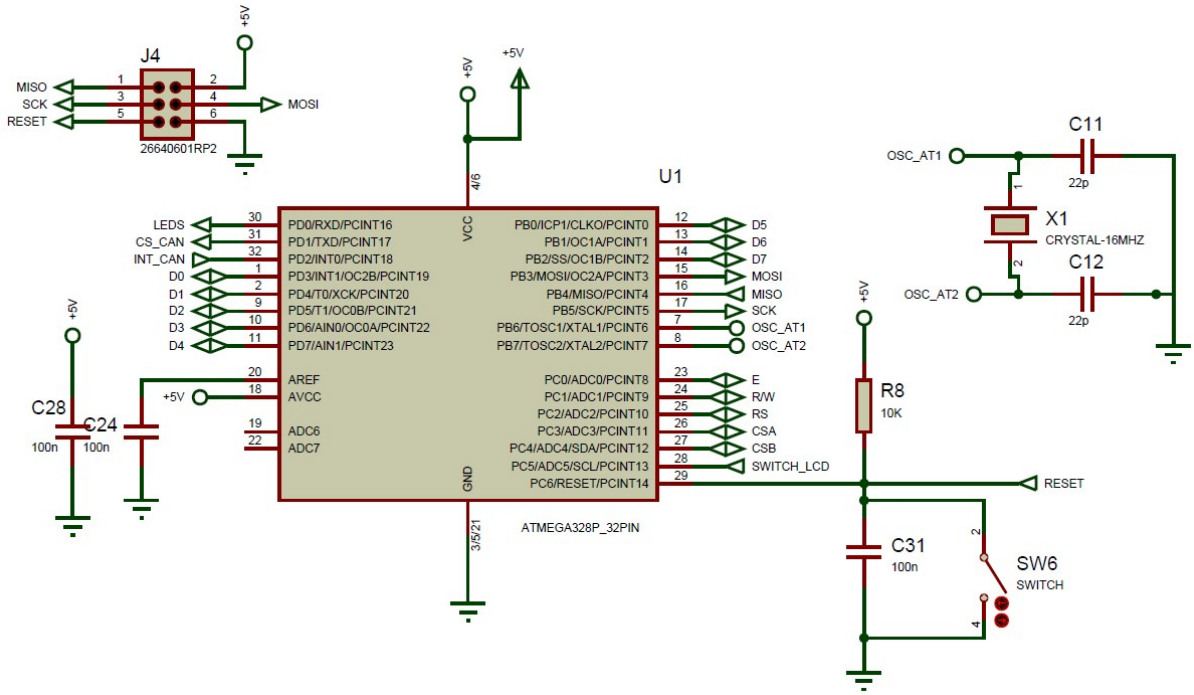

Figura 47: Diseño esquemático. Microcontrolador y elementos auxiliares.

Alimentación

Como puede observarse hay cuatro pines del convertidor DC/DC conectados, esto se debe al anteriormente comentado aislamiento. A él llega la tensión de la batería y la convierte a los 5V de tensión de alimentación de los distintos componentes del sistema. El aislamiento resulta de gran interés debido a que impide la entrada de ruido a través de la alimentación, separando eléctricamente la entrada de la salida.

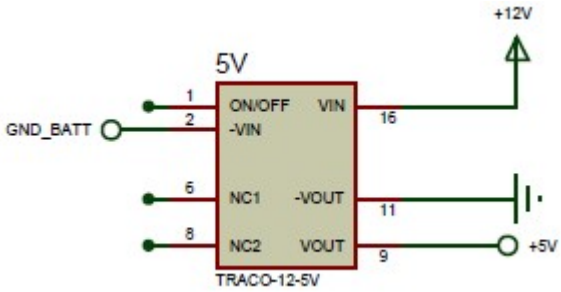

Figura 48: Diseño esquemático. Alimentación.

Bus CAN

Para implementar Bus CAN en el sistema es son necesarios varios componentes, como ya se comentó en el apartado anterior. Son necesarios, el controlador y el transceiver, además de las dos posibles formas de resistencia terminal del bus, la configuración Estándar y la configuración Split.

El esquemático perteneciente al transceiver se muestra en la Figura 49, aquí se puede observar que el único elemento auxiliar es la resistencia que se coloca en el pin "Rs" del chip, con esta entrada decimos al transceiver el modo de funcionamiento, en este caso al poner la resistencia de pull down estamos usado el modo "slope control mode".

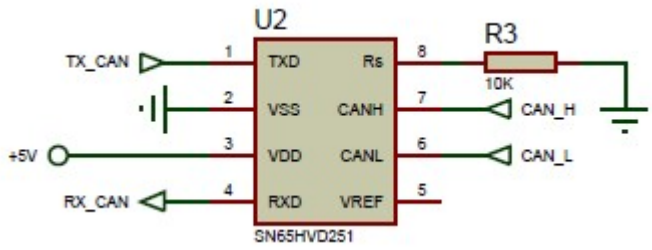

Figura 49: Diseño esquemático. Transceiver CAN.

En la Figura 50 se muestra el controlador CAN junto con componentes auxiliares; condensador de alimentación, oscilador y resistencia de pull up en la señal de reset.

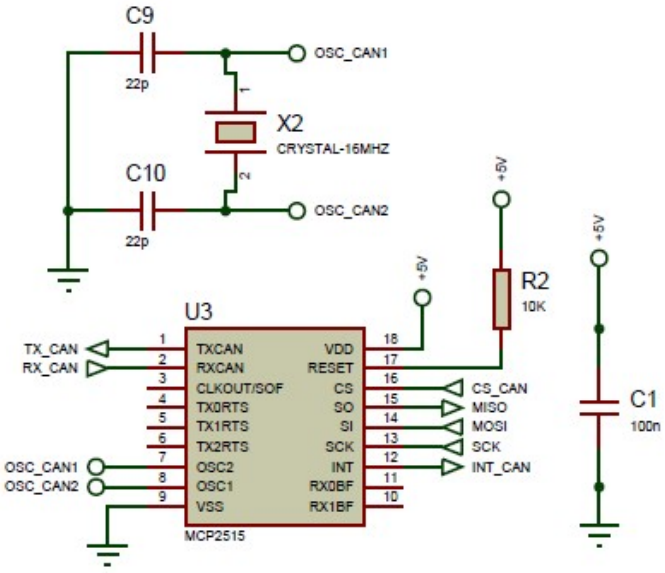

Figura 50: Diseño esquemático. Controlador CAN y elementos auxiliares.

En el diseño se incluyen las dos configuraciones de resistencia terminal, esto se incluye en todos los sistemas, para dar mayor flexibilidad a cada sistema. Estas resistencias se conectan al bus mediante jumpers por lo que se puede cambiar de una configuración a otra fácilmente, e inluso no poner ninguna ya que solo es necesario poner esta resistencia en los extremos del bus. En la Figura 51 se muestra el diseño esquemático.

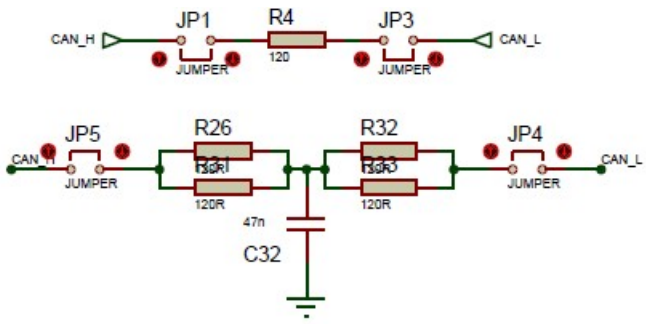

Figura 51: Diseño esquemático. Resistencias terminales Bus CAN.

LEDs: revoluciones y alertas

Los LEDs tanto de revoluciones como los de alarma se colocan en serie. Siguiendo la recomendación del datasheet se ha colocado un condensador de 100nF entre alimentación y tierra y una resistencia de 500Ω en la entrada al primer LED para evitar daños en éste.

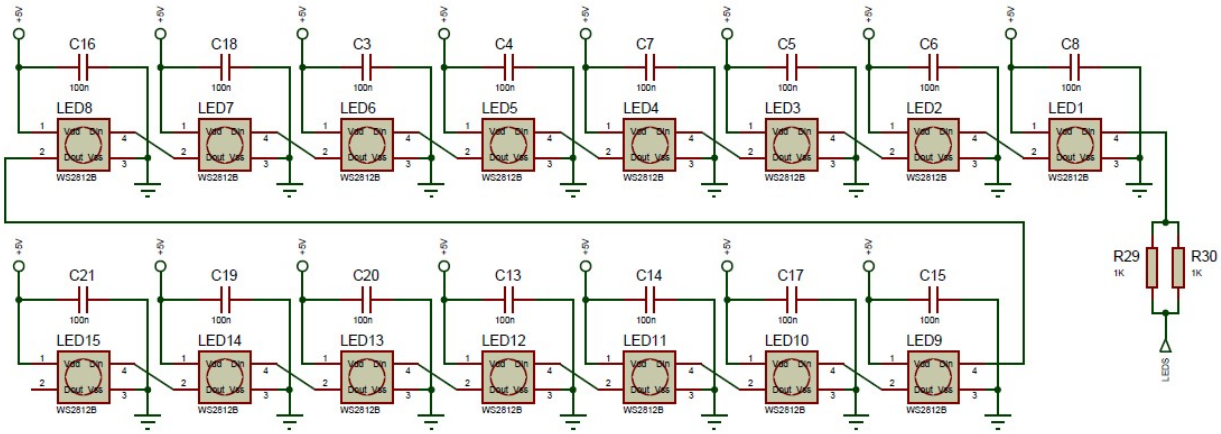

Figura 52: Diseño esquemático. LEDs de revoluciones y de alarma.

Interruptores

Como se comentó, era requisito la presencia de interruptores en el diseño para interactuar con el propio sistema y con otros sistemas. En la Figura 53 se muestra el diseño esquemático de estos, todos ellos son conmutadores. De izquierda a derecha; cambio de mapa en la ECU, activación manual de la refrigeración del motor, activación de la ADQ y cambio de la configuración de la pantalla. Todos ellos a excepción de la señal que va a la ECU, la cual tiene una resistencia de pull up interna, tienen su respectiva resistencia de pull up a la tensión de alimentación del sistema al que atacan.

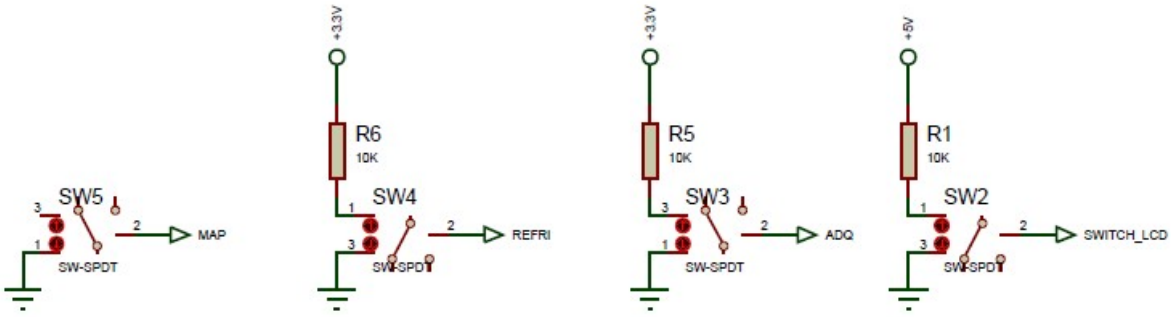

Figura 53: Diseño esquemático. Interruptores.

Conectores

En el diseño hay varios conectores, dos de ellos solo tienen la finalidad de pasar señales al conector general, señales provenientes del volante y de los botones de contacto y arranque. En las siguientes Figuras se muestran estos conectores.

Figura 54: Diseño esquemático. Conector volante (izq) y botones (der).

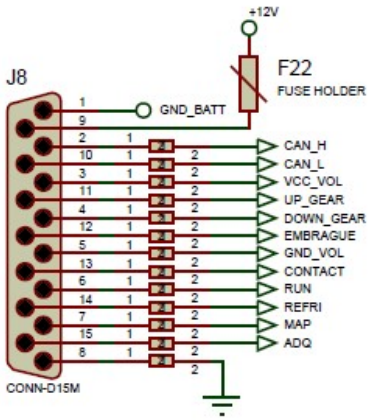

Figura 55: Diseño esquemático. Conector general.

Por último, el conector de la pantalla, el cual se muestra en la Figura 56 y junto al que se muestran los elementos auxiliares; el potenciomentro para regular el contraste de la misma y un jumper que permite poner la señal de reset a alimetación o a tierra.

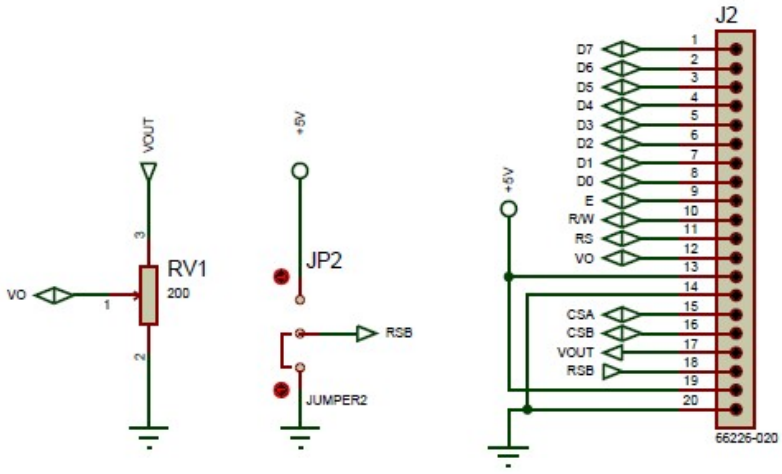

Figura 56: Diseño esquemático. Conector pantalla y elementos auxiliares.

APÉNDICE B. LAYOUT

En las siguientes Figuras se muestra el layout de la PCB, donde se pueden observar las capas Top, Botton y el layout completo. Nótese que en este layout también está o relativo al otro sistema que está integrado en esta PCB, la Telemetría del vehículo.

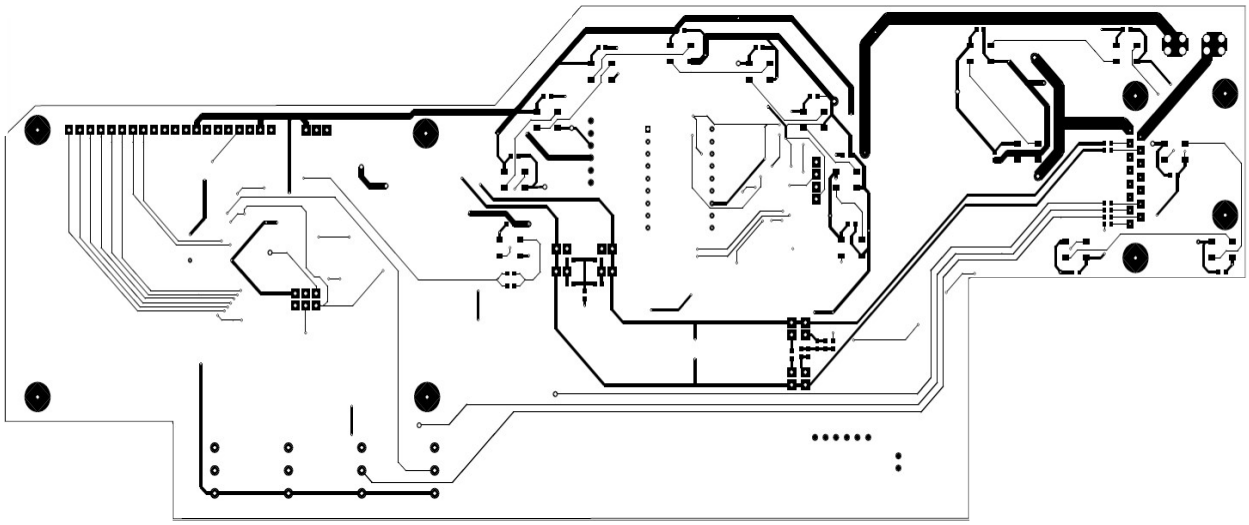

Figura 57: Layout capa Top.

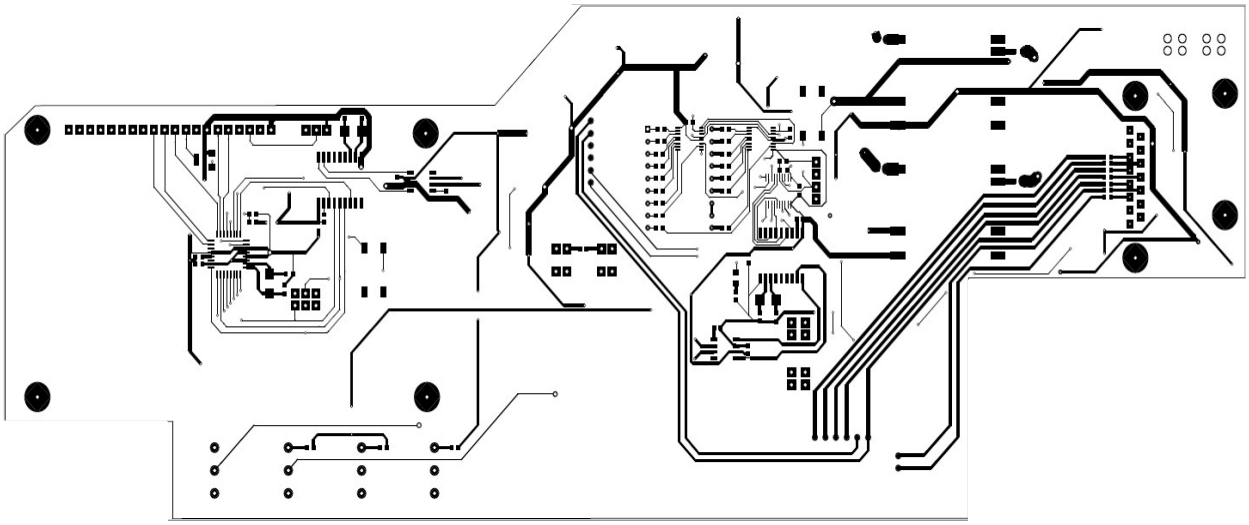

Figura 58: Layout capa Botton.

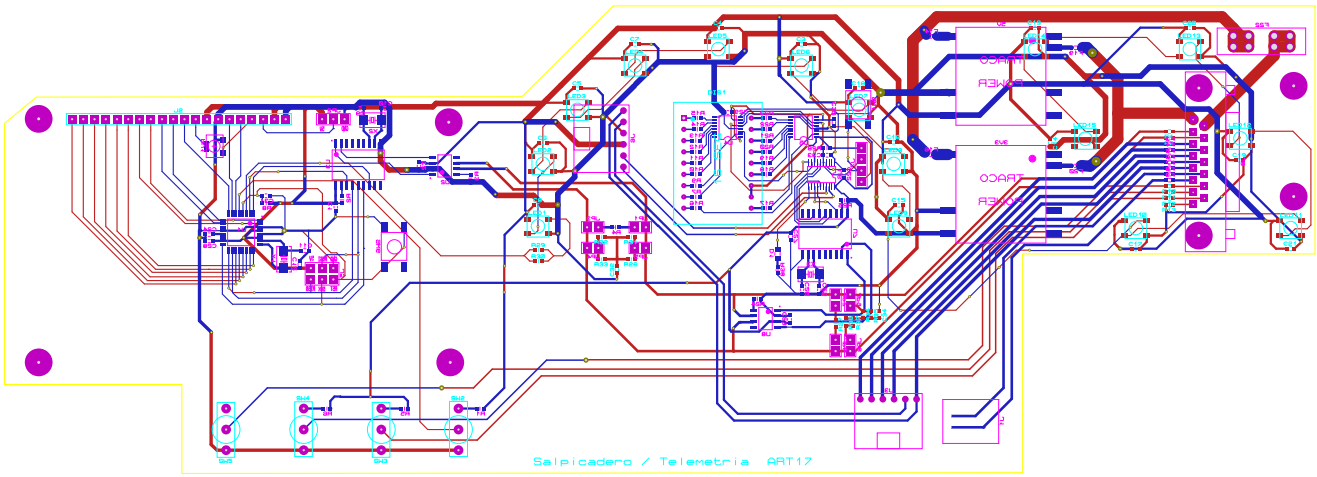

Figura 59: Layout completo.

APÉNDICE C. CÓDIGO FUENTE

Programa principal

```
#include <mcp_can.h>
#include <SPI.h>
#include "U8glib.h"
#include <Adafruit_NeoPixel.h>
#include "icons.h"
#include "defs.h"
#include <TimerOne.h>

U8GLIB_KS0108_192 u8g(3, 4, 5, 6, 7, 8, 9, 10, A0, A3, A4, A2, A1);
//*** D0.D1.D2.D3.D4.D5.D6.D7.E.CS1.CS2.RS.RW ***//

const int SPI_CS_PIN = 1;
MCP_CAN CAN(SPI_CS_PIN);

#define PIN            0
#define NUMPIXELS      15
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB +
NEO_KHZ800);

unsigned char Id;

unsigned int rev = 0;                       // valor de las revoluciones
int delayval = 0;                 //tiempo de parpadeo en revoluciones al
máximo
int lambda = 0;
int lambda_u = 0;
int lambda_d = 0;

int oil_pres_int = 0;
int oil_pres_u = 0;
int oil_pres_d = 0;
float oil_pres = 5.0;
int oil_T = 50;
int a = 0;
int b = 0;
int cockpit_T = 25;
int batt = 0;
int water_T = 50;

char record = 0;
//unsigned char Id_Error = 0x0A;

unsigned char flagRecv = 0;
unsigned char len = 0;
unsigned char buf[8];

uint32_t color[15] = {pixels.Color(0, 250, 0), pixels.Color(0, 200, 0),
pixels.Color(0, 150, 0),
                      pixels.Color(150, 200, 0), pixels.Color(150, 150, 0),
pixels.Color(200, 150, 0),
                      pixels.Color(200, 50, 0), pixels.Color(200, 0, 0),
pixels.Color(255, 0, 0),

                      pixels.Color(255, 0, 0), pixels.Color(255, 0, 0),
                      pixels.Color(255, 0, 0), pixels.Color(255, 0, 0),
```

```
                    pixels.Color(255, 0, 0), pixels.Color(255, 0, 0)
                  };


bool refresh = true;
bool errorPot = false;
char codeErrorPot = 0;
bool errorADQ = false;
char codeErrorADQ = 0;
int timer = 0;
bool ledrev = false;
bool aux = false;

int pot = 0;
bool normal_function = true;
bool no_data = false;
unsigned long t_last_data = 0;

char texto[20];
char waterTString[6] = {'5', '0', '°'};
char oilTString[5] = {'9', '5', '°'};
char cockpitString[5] = {'3', '3', '°'};
char oilPressString[] = "8.022 bar";
char batteryString[5] = "99%";
char lambdaString[8] = "0.9";
char revString[7] = "12000";


void u8g_prepare(void) {
  u8g.begin();
  u8g.setFont(u8g_font_fub20);
  u8g.setFontRefHeightExtendedText();
  u8g.setDefaultForegroundColor();
  u8g.setFontPosTop();
}

void draw(void) {
  u8g.setFont(u8g_font_fub20);
  pantalla1();
  switch (digitalRead(A5)) {
    case true:
      pantalla1();
      break;
    case false:
      pantalla2();
      break;
  }
}

void drawd(void) {
  if (oil_T > 99) {
    u8g.drawStr( 10, 40, oilTString);
  }
  else {
    u8g.drawStr( 21, 40, oilTString);
  }
  if (water_T > 99) {
    u8g.drawStr( 70, 40, waterTString);
  }
  else {
    u8g.drawStr( 85, 40, waterTString);
  }
```

```
  u8g.setFont(u8g_font_timB14);
  u8g.drawStr(35, 61, oilPressString);
  u8g.drawStr(155, 18, cockpitString);
  u8g.drawStr(155, 40, batteryString);
  u8g.drawStr(150, 61, lambdaString);

  u8g.drawBitmapP(0, 0, 24, 92, idisp);
}

void drawError(void) {
  u8g.setFont(u8g_font_timB14);
  if (errorPot == true) {
    switch (codeErrorPot) {
      case 0x10:
        u8g.drawStr(0, 20, "BATERIA BAJA");
        break;
      case 0x20:
        u8g.drawStr(0, 20, "Bateria Recargada");
        break;
      case 0x30:
        u8g.drawStr(45, 20, "APAGADO");
        u8g.drawStr(35, 40, "DE SISTEMAS");
        u8g.drawStr(20, 60, "PRESCINDIBLES");
        break;
    }
  }
  else if (errorADQ == true) {
    switch (codeErrorADQ) {
      case 0x10:
        u8g.drawStr(0, 20, "No se pudo");
        u8g.drawStr(0, 40, "montar la");
        u8g.drawStr(0, 60, "unidad virtual");
        break;
      case 0x20:
        u8g.drawStr(0, 40, "Fallo LOG");
        break;
      case 0x30:
        u8g.drawStr(0, 40, "PRESION DE ACEITE");
        break;
      case 0x40:
        u8g.drawStr(0, 40, "TEMPERATURA");
        break;
      case 0x50:
        u8g.drawStr(0, 40, "PRESION DE FRENADA");
        break;
    }
  }
}

void draw_noData(void) {
  u8g.setFont(u8g_font_timB14);
  u8g.drawStr(40, 40, "NO DATA !");
}

void draw_test(void) {
  u8g.setFont(u8g_font_timB14);
  u8g.drawStr(25, 15, "ECT");
  u8g.drawStr(105, 15, "Lambda");
  u8g.setFont(u8g_font_fub20);
  u8g.drawStr(105, 40, lambdaString);
  u8g.drawStr( 25, 40, waterTString);
```

```
  u8g.drawStr( 55, 63, revString);
}

void draw_test2(void) {
  u8g.setFont(u8g_font_timB14);
  u8g.drawStr(25, 15, "Oil T");
  u8g.drawStr(105, 15, "Oil P");
  u8g.setFont(u8g_font_fub20);
  u8g.drawStr(105, 50, oilPressString);
  u8g.drawStr( 25, 50, oilTString);
}

void pantalla1(void) {
  if ((errorPot == true || errorADQ == true) && aux) // esto se hace para que
parpadee
    drawError();
  else
    drawd();
}

void pantalla2(void) {
  sprintf(texto, "Id recibida: %d", Id);
  u8g.drawStr(10, 30, texto);
}



void revolutions(int in) {
  in = min(in, 9);
  for (int i = 0; i <= in - 1; i++) {
    pixels.setPixelColor(i, color[i]);
    pixels.show();
  }

  for (int i = in; i <= 8; i++) {
    pixels.setPixelColor(i, pixels.Color(0, 0, 0));
    pixels.show();
  }
}

void led_adv(void) {
  if (record >= 100) {
    pixels.setPixelColor(10, pixels.Color(0, 150, 0));
    pixels.show();
  }
  else {
    pixels.setPixelColor(10, pixels.Color(0, 0, 0));
    pixels.show();
  }

  if (water_T > 104) {
    pixels.setPixelColor(14, color[10]);
    pixels.show();
  }
  else if (water_T > 94) {
    pixels.setPixelColor(14, pixels.Color(250, 150, 0));
    pixels.show();
  }
  else if (water_T > 70) {
    pixels.setPixelColor(14, pixels.Color(0, 150, 0));
    pixels.show();
  }
```

```
  else {
    pixels.setPixelColor(14, pixels.Color(0, 0, 0));
    pixels.show();
  }

  if (oil_T > 104) {
    pixels.setPixelColor(13, color[10]);
    pixels.show();
  }
  else if (oil_T > 96) {
    pixels.setPixelColor(13, pixels.Color(250, 150, 0));
    pixels.show();
  }
  else if (oil_T > 70) {
    pixels.setPixelColor(13, pixels.Color(0, 150, 0));
    pixels.show();
  }
  else {
    pixels.setPixelColor(13, pixels.Color(0, 0, 0));
    pixels.show();
  }

  if (oil_pres_int < 50) {
    pixels.setPixelColor(12, color[10]);
    pixels.show();
  }
  else if (oil_pres_int < 80) {
    pixels.setPixelColor(12, pixels.Color(250, 150, 0));
    pixels.show();
  }
  else {
    pixels.setPixelColor(12, pixels.Color(0, 0, 0));
    pixels.show();
  }

  if (cockpit_T > 60) {
    pixels.setPixelColor(11, color[10]);
    pixels.show();
  }
  else if (cockpit_T > 40) {
    pixels.setPixelColor(11, pixels.Color(250, 150, 0));
    pixels.show();
  }
  else {
    pixels.setPixelColor(11, pixels.Color(0, 0, 0));
    pixels.show();
  }
}

void timerIsr() {
  timer++;
  if (timer == 50) {
    refresh = true;
    timer = 0;
  }
}

void setup(void) {

  pinMode(A5, INPUT);

  Timer1.initialize(50000); //50ms
```

```
  Timer1.attachInterrupt( timerIsr ); // attach the service routine here

  u8g_prepare();
  pixels.begin();
  pixels.setBrightness(75);

  revolutions(10);
  delay(500);

  while (CAN_OK != CAN.begin(CAN_500KBPS))
  {
    pixels.setPixelColor(0, pixels.Color(150, 0, 0));
    pixels.show();
    delay(100);
  }


  CAN.init_Mask(0, 0, 0xFF);
  CAN.init_Mask(1, 0, 0xFF);

  CAN.init_Filt(0, 0, 0x46);
  CAN.init_Filt(1, 0, 0x46);

  CAN.init_Filt(2, 0, 0x14);
  CAN.init_Filt(3, 0, 0x14);
  CAN.init_Filt(4, 0, 0x14);
  CAN.init_Filt(5, 0, 0x14);

  revolutions(0);
  //attachInterrupt(0, MCP2515_ISR, FALLING);
}


bool datoECU = false;
bool datoADQ = false;


void loop(void) {

  revolutions(rev);
  led_adv();

  if (CAN_MSGAVAIL == CAN.checkReceive()) {
    CAN.readMsgBuf(&len, buf);
    Id = CAN.getCanId();

  }
  if (Id == Id_ADQ) {

    datoADQ = true;
    record = buf[p_record];
    // Descomentar para que la pantalla se actualice cuando llegan datos de
la ADQ
**************************************************************************
***
    //Oil Tª
    oil_T = buf[p_oil_T];
    //Cockpit Tª
    cockpit_T = buf[p_cockpit_T];
    //Battery
    batt = buf[p_batt];
```

```
}
else if (Id == Id_ECU1) {

  datoECU = true;
  oil_pres_int = (buf[p_oil_pres_MSB] << 8) | buf[p_oil_pres_LSB];
  // oil_pres_u = oil_pres_int / 1000;
  // oil_pres_d = oil_pres_int % 1000;

  rev = (buf[p_rev_MSB] << 8) | buf[p_rev_LSB];
  rev = map(rev, 0, 13000, 0, 10);
  lambda = (buf[p_lambda_u] << 8) | buf[p_lambda_d];
  water_T = buf[p_water_T];
  lambda_u = lambda / 1000;
  lambda_d = lambda % 1000;


}
else if (Id == Id_Error) {
  if (buf[0] == 0 && buf[1] != 0) {
    errorPot = true;
    codeErrorPot = buf[1];
  }
  else if (buf[0] == 0 && buf[1] == 0) {
    errorPot = false;
  }
  else if (buf[0] == 255 && buf[1] != 0) {
    errorADQ = true;
    codeErrorADQ = buf[1];
  }
  else if (buf[0] == 0 && buf[1] == 0) {
    errorADQ = false;
  }
}


if (refresh == true) {
  refresh = false;

  if (datoECU == true) {
    datoECU = false;
    sprintf(lambdaString, "%d.%d", lambda_u, lambda_d);
    sprintf(waterTString, "%d%c", water_T, '°');
    sprintf(oilPressString, "%d kPa", oil_pres_int);
  }
  else {
    sprintf(lambdaString, " -- ");
    sprintf(waterTString, " -- ");
    sprintf(oilPressString, " -- ");
  }

  if (datoADQ == true) {
    datoADQ = false;
    //sprintf(oilTString, "%d%c", oil_T, '°');
    sprintf(oilTString, " -- ");
    sprintf(cockpitString, "%d%c", cockpit_T, '°');

    // oil_pres = oil_pres_u + 0.1*oil_pres_d;
    //sprintf(batteryString, "%d%c", batt, '%');
    sprintf(batteryString, " -- ");
  }
  else {
    sprintf(oilTString, " -- ");
```

```
      sprintf(cockpitString, " --");
      sprintf(oilPressString, " -- ");
      sprintf(batteryString, " -- ");
    }

    u8g.firstPage();
    do {
      draw();
    } while ( u8g.nextPage() );
  }
  Id = 0;

}
```

Definiciónes

```
/* ECU */
#define Id_ECU1 0x46
#define p_rev_LSB 1
#define p_rev_MSB 0
#define p_lambda_u 3
#define p_lambda_d 4
#define p_water_T 2
#define p_oil_pres_LSB 7
#define p_oil_pres_MSB 6

/* ADQ */
#define Id_ADQ 0x14
#define p_oil_T 1
#define p_record 0
#define p_cockpit_T 2
#define p_batt 6
```

Gráficos

```
const uint8_t idisp[] PROGMEM = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x10, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x19, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1D, 0xF0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x3E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x2E, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x1E, 0xF0, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3E, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x28, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x16, 0xE1, 0xE0, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x2E, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x13, 0xF3, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3E, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
  0xC0, 0x28, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x13, 0xFF, 0xF0, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x3E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x2E, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x1B, 0xFF, 0xB0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x28, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0xFF, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x3E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x28, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x0F, 0x03, 0x90, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3E, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x28, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x03, 0x38, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x28, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x03, 0x07, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x28, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x06, 0x7C, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x1F, 0x7B, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x6C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x03, 0x06, 0x7C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x03, 0xF0, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0xC6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xFE, 0x7C, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x3B, 0xFF, 0x70, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x82, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x03, 0xFE, 0x7C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xFF, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0xC6, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xFC, 0x38, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x6C, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x03, 0xFC, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xFF, 0xF8, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x3F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC3, 0xC3, 0xC0, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC3, 0xC3, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xDF, 0xFF, 0xF0, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xDF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xD8, 0x00, 0x30, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xD9, 0x80, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xDB, 0xC3, 0xB0, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xD9, 0x80, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xD8, 0x00, 0x30, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xD8, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xD8, 0x00, 0x30, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xD8, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xD8, 0x00, 0x30, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xDF, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xDF, 0xFF, 0xF0, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF,
  0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF,
  0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xC0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC1, 0xE0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC3, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC3, 0x30, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
  0x00, 0xC1, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC3, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC1, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x30, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x01, 0xC9, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x9C, 0xE0, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x38, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x13, 0x3C, 0xE4, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7B, 0x36, 0x6F, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x38, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x7F, 0x63, 0x7F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7B, 0x63, 0x6F, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x78, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x13, 0x63, 0x64, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0xEC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x3E, 0x60, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xEC, 0xC0, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x03, 0x9C, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC1, 0xCC, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xC1, 0xC0, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC1, 0x8F, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC3, 0x87, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3E, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC3, 0x03, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00
};
```

APÉNDICE D. BUS CAN

A lo largo del proyecto se ha mencionado en numerosas ocasiones la utilización del bus CAN. Para tener un conocimiento global de esta tecnología se ha llevado a cabo estudio de las características, componentes y forma de envío del bus.

Bus CAN (Controller Area Network) es un protocol de comunicación en serie desarrollado por Bosch en los años 80 para el intercambio de información entre unidades de control electrónicas del automóvil, aunque actualmente se utiliza en otros sectores del área de control y automatización industrial. La robustez del Bus CAN se basa en su arquitectura multimaestro. Este sistema permite compartir una gran cantidad de información entre los diferentes módulos de control conectados a la red, lo que produce una reducción importante tanto del número de sensores utilizados como de la cantidad de cables que componen la instalación electrica. De esta forma aumentan considerablemente las funciones actuales en los sistemas del automóvil donde se emplea Bus CAN sin aumentar los constes, además de que estas funciones pueden estar repartidas entre dichos módulos de control.

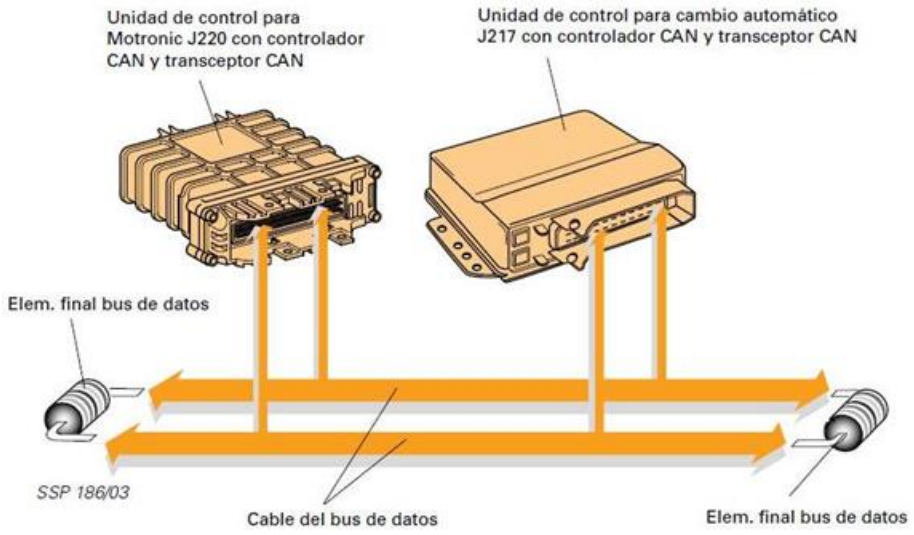

Figura 60: Modulos conectados a una red CAN.

Características principales Bus CAN

- La información que circula entre los dispositivos que implementan CAN a través de los cables (bus) son paquetes de bits con una longitud limitada y con una estructura definida en campos.
- Uno de los campos actúa de identificador del tipo de datos que se transporta, de la unidad de mando que lo transmite y de la prioridad para transmitirlo resoecto a otros. El mensaje no va direccionado a ninguna unidad de mando en concreto, sino que cada una de ella reconocerá mediante este identificador el mensaje que le interesa.
- Todas las unidades de la red pueden ser transmisoras y receptoras, y el numero de módulos conectados a la red puede ser variable.
- Cualquier dispositivo introduce un mensaje en el bus con la condición de que esté libre, si otro lo intenta al mismo tiempo el conflicto se resuelve por la prioridad del mensaje enviado con el identificador más bajo.
- El sistema está dotado de una serie de mecanismos que aseguran que el mensaje es transmitido y recibido correctamente. Cuando un mensaje presenta un error, es anulado y vuelto a transmitir de forma correcta (automáticamente).

Protocolo de enlace

Consta de un gran número de bits. La cantidad de bits de un protocolo depende del tamaño del campo de datos, el cual se indica en un campo fijo de la trama. En la Figura 61 se muestra la estructura de una trama CAN.

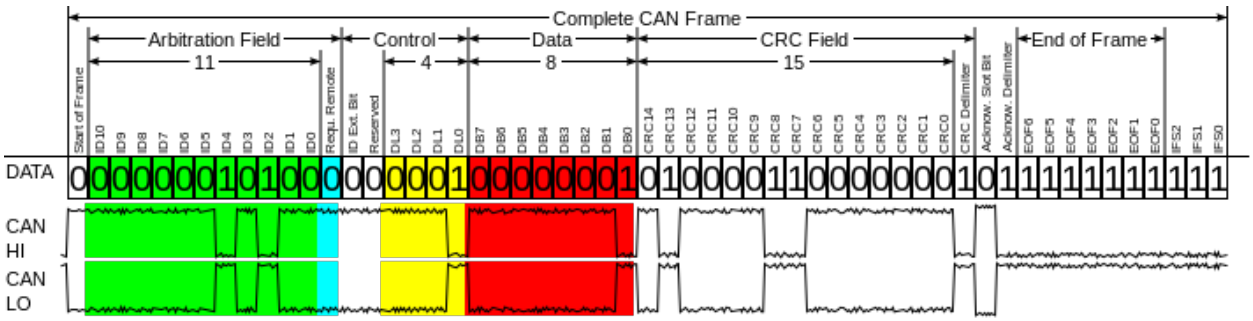

Figura 61: Trama CAN.

- El campo *Start of frame* da comienzo a la trama y permite la sincronización de los nodos. Tiene estado dominante.
- El campo *Árbitro* está compuesto por entre 12 o 32 bits. Dentro de este campo se encuentra el identificador. El bir *RTR* distingue entre una trama de datos y una trama remota.
- El campo de *Control* está formado por 6 bits que contienen información del protocolo.
- El campo de *Datos* puede estar formado hasta por 8 bytes. Contiene los datos del mensaje.
- El campo CRC se compone de 15 bits y sirve para la detección de errores en el mensaje.
- El campo *Fin de trama* está compuesto por 7 bits recesivos que indican el final de la trama de datos.

Capas

- La capa física es la responsable de la transferencia de bits entre los distintos módulos que forman la red. Define aspectos a nivel de señal, codificación, sincronización y tiempos en que los bits se transfieren al bus.
- La capa de enlace define el método de acceso al medio utilizado en CAN. Resuelve la colisión de varias tramas, con la supervivencia de la trama cuyo identificador tenga mayor prioridad.

Elementos que componen el Bus CAN

- La información circula por dos cables trenzados que unen todos los módulos que forman el sistema. Esta información se transmite por diferencia de tensión entre los dos cables, de forma que un valor alto de tensión representa un 1 y un valor bajo de tensión representa un 0. La combinación adecuada de unos y ceros forman el mensaje a transmitir. En un cable los valores de tensión oscilan entre 0V y 2.25V, por lo que se denomina cable L (Low) y en el otro, el cable H (High) lo hacen entre 2.75V. y 5V. En caso de que se interrumpa la línea H o que se derive a masa, el sistema trabajará con la señal de Low con respecto a masa, en el caso de que se interrumpa la línea L, ocurrirá lo contrario. Esta situación permite que el sistema siga trabajando con uno de los cables cortados o comunicados a masa, incluso con ambos comunicados también sería posible el funcionamiento, quedando fuera de servicio solamente cuando los dos cables se cortan. Es importante tener en cuenta que el trenzado entre ambas líneas sirve para anular los campos magnéticos, por lo que no se debe modificar en ningún caso ni el paso ni la longitud de dichos cables.

- Como elemento terminal se coloca una resistencia conectada en los extremos de los cables H y L. Sus valores son habitualmente 120 Ohmios (aunque pueden variar) y se colocan para adecuar el funcionamiento del sistema a diferentes longitudes de cables y número de módulos conectados, ya que impiden posibles efectos parásitos que nos pueden trastocar el mensaje. Estas resistencias las colocamos en la mismísima placa del módulo, para así ahorrar en soluciones externas y para mayor seguridad de funcionamiento.

- El controlador es el elemento encargado de la comunicación entre el microprocesador del módulo y el trasmisor-receptor. Trabaja acondicionando la información que entra y sale entre ambos componentes. Habrá un controlador por cada módulo de nuestra red. Este elemento trabaja con niveles de tensión muy bajos y es el que determina la velocidad de trasmisión de los mensajes, que será más o menos elevada dependiendo de nuestra aplicación y lo que esperemos de ella. El controlador también interviene en la sincronización entre los diferentes módulos para la correcta emisión y recepción de los mensajes.

- El transceptor o transceiver es el elemento que tiene la misión de recibir y de trasmitir los datos, además de acondicionar y preparar la información para que pueda ser utilizada por los controladores. Esta preparación consiste en situar los niveles de tensión de forma adecuada, amplificando la señal cuando la información se vuelca en la línea y reduciéndola cuando es recogida de la misma y suministrada al controlador. El transceptor es básicamente un circuito integrado que está situado en cada uno de los módulos, trabaja con intensidades próximas a 0.5 A y en ningún caso interviene modificando el contenido del mensaje. Para su buen funcionamiento se sitúa entre los cables que forman la línea CAN Bus y el controlador.

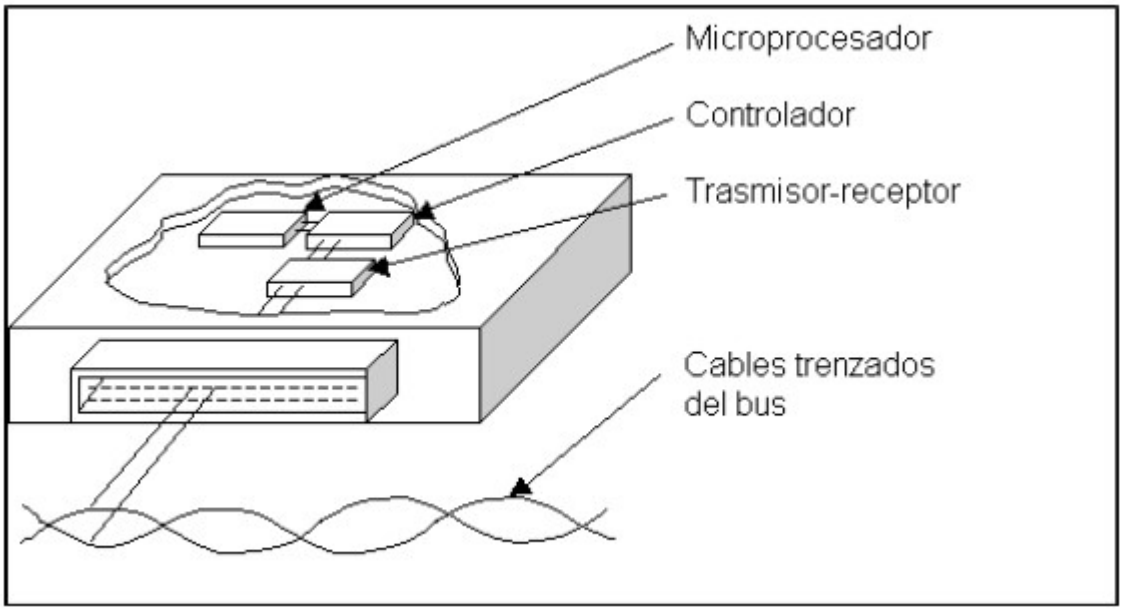

Figura 62: Nodo CAN.

Ejemplo

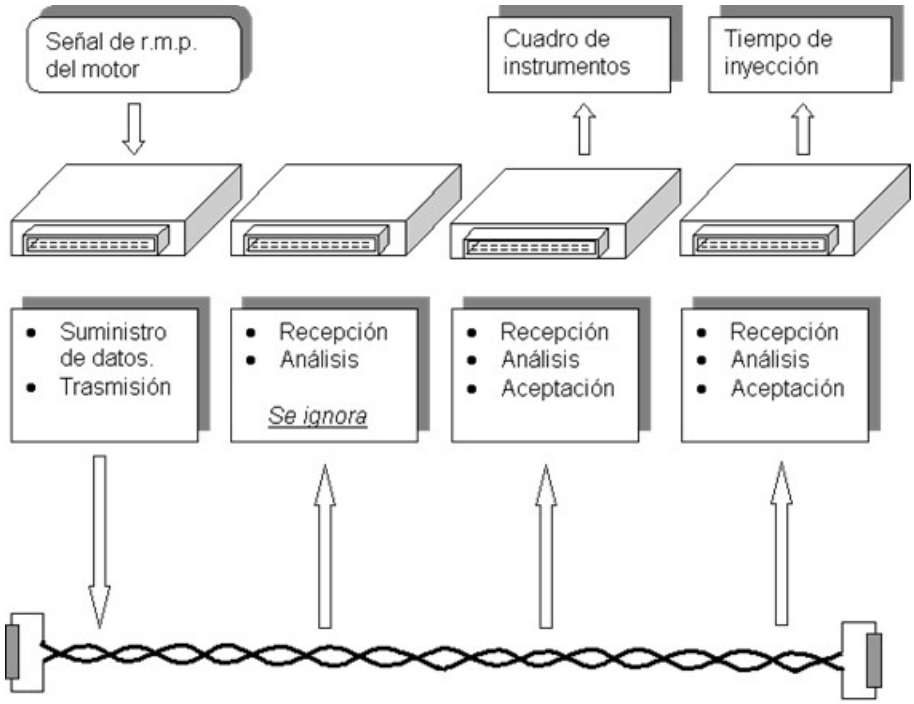

Figura 63: Ejemplo práctico Bus CAN.

En el caso de que uno o varios módulos quieran introducir un mensaje al mismo tiempo, lo hará la que tenga una mayor prioridad. Esta prioridad nos la indica el identificador. El proceso de trasmisión de datos se desarrolla siguiendo varios pasos:

- **Suministro de datos**. Un módulo recibe información de los sensores que tiene asociados (RPM de un motor, velocidad, temperatura del motor en nuestro caso, etc) Su microprocesador pasa la información al controlador donde es gestionada y preparada para a su vez ser pasada al transceptor donde se traducirá en señales eléctricas.
- **Transmisión de datos**. El controlador de dicho módulo transfiere los datos y el identificador junto con la petición de inicio de trasmisión, asumiendo la responsabilidad de que el mensaje sea correctamente transmitido a todos los módulos de la red. Para transmitir el mensaje ha tenido que encontrar el bus libre, y en caso de conflicto con otro módulo intentando transmitir al mismo tiempo, tener una prioridad mayor. A partir del momento en que esto ocurre, el resto de módulos se convierten en receptores.
- **Recepción del mensaje**. Cuando todos los módulos reciben el mensaje, verifican el identificador para determinar si el mensaje va a ser utilizado por ellos. Los módulos que necesiten los datos del mensaje lo procesan, si no lo necesitan, el mensaje se ignora. El sistema bus CAN dispone de mecanismos para detectar errores en la trasmisión de mensajes, de forma que todos los receptores realizan un chequeo del mensaje analizando el campo CRC. El planteamiento del Bus CAN, como puede observarse, permite disminuir notablemente el cableado de nuestra utilidad, puesto que, si un módulo dispone de una información, como, por ejemplo, la temperatura de aceite, esta puede ser utilizada por el resto de módulos sin que sea necesario que cada uno de ellos reciba la información de ese sensor de temperatura.

REFERENCIAS

[1] FSAE, 2017 Formula SAE® Rules, SAE International, 2017. http://www.fsaeonline.com/content/2017-18%20FSAE%20Rules%20PRELIMINARY.pdf

[2] Rua Copeto D. Automotive data acquisition system – FST. Insituto Superior Técnico. Universidade Técnica de Lisboa. 2009.

[3] Mancini., B Greene B., Fiederlein E., Doelger G and Tung K. The Complete Design of an Engine Management System for RIT's Formula SAE Team. Multi-Disciplinary Engineering.

[4] López Fresno J., Nodo de comunicación basado en el BUS CAN. PFC. Universidad Rovira i Virgili. 2004.

[5] Alejandro García Osés. Diseño de una red CAN bus con Arduino. Universidad Politécnica de Navarra.

[6] Francisco Jesús Matas Díaz. Gestión del sistema eléctrico de un monoplaza. Universidad de Sevilla.

[7] Daniel Gómez Sanz. Diseño de un sistema de telemetría para un monoplaza de Formula Student. Universidad de Sevilla.

[8] Fecha de consulta: 5 de octubre 2016. https://es.wikipedia.org/wiki/Bus_CAN

[9] Fecha de consulta: 5 de octubre 2016. http://www.aficionadosalamecanica.net/canbus.htm

[10] Fecha de consulta 20 de abril de 2017. www.formulastudent.de/fileadmin/user_upload/all/2017/rules/FS-Rules_2017_V1.1.pdf

[11] SAE-International. Competition History. http://www.sae.org/ students/ fsaehistory.pdf , April 2005. Fecha de consulta: 1 de diciembre 2016.

[12] Fecha de consulta: 22 de Nobiembre 2016. https://www.arduino.cc/ en/ Main/ ArduinoBoardUno

[13] Fecha de consulta: 22 de Nobiembre 2016. http://www.seeedstudio.com/ wiki/CAN-BUS_Shield

[14] Fecha de consulta: 22 de Nobiembre 2016. https://www.arduino.cc/ en/ Main/ ArduinoBoardUno

[15] Fecha de consulta: 10 de Febrero 2017. http://www.linkecu.com/

[16] Fecha de consulta: 10 de Febrero 2017. http://www.linkecu.com/ Software-and-Support/PC-Link-Downloads